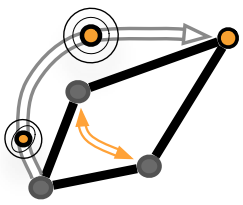
Université de Strasbourg

Master de Science, mention Informatique,
spécialité Réseaux Informatiques et Systèmes Embarqués

Presented by:
Jean-Romain Luttringer
jr.luttringer@unistra.fr

# OPTIC: An Efficient BGP Protection Technique For Optimal Intra-domain Convergence

Supervised by:
Pascal Mérindol
merindol@unistra.fr

NETWORK RESEARCH TEAM

February-August 2019

# Acknowledgements

While I would love to be able to brag with the fact that I did this work by myself, it could not be further from the truth. Consequently, there are quite a few people I would like to thank.

First and foremost, I would, of course, like to thank Pascal Mérindol for his time, effort, and support during this whole internship. I would also like to apologize (kind of) for ruining his holiday with my never-ending stream of messages, which were every time answered with wise and useful advice. Besides, I cannot deny that one of my main motivations and one of the main causes behind most of my contributions was my uncontrollable desire to prove him wrong and/or to beat him to the punch as many times as possible, which I sadly wasn't able to do as many times as I would have liked.

I would also like to thank Cristel Pelsser for taking the time to proofread this thesis and for always being available upon my many requests for advice. Once again, I would like to apologize in advance for the work I will add to her pile as her Ph.D. student.

As these acknowledgments are already becoming quite long, I will wrap this section up by thanking as a whole the networking team, permanent members and Ph.D. students alike, for their warm welcome and for making the laboratory a nice and friendly workplace. At the risk of making some people jealous, I would like to thank especially Julian Martin Del Fiore, who managed to read this report and gave me numerous writing advice (most of them being actually very useful!).

# Table of contents

# List of Figures

# Chapter 1 Introduction

## 1. Hosting Structure

My internship was conducted within the Network Research Team of the ICube laboratory. The ICube laboratory, created in 2013, currently hosts around 650 members, working in 16 research teams, whose domains range from fluid dynamics to computer science. Hosting two distinct scientific communities is one of ICube's strong suit, as it allows a strong cooperation and communication between the digital and physical world. Indeed, being a joint laboratory regrouping four entities (the University of Strasbourg, the National Center for Scientific Research, the National School for Water Environmental Engineering and the National Institute of Applied Science), ICube hosts researchers and experts from a vast array of fields. In addition, ICube houses two well-known platforms, Robotex and FIT, which are respectively dedicated to medical robotics and the Internet of Things. ICube is a close partner of several other scientific actors, such as INRIA, the Faculty of Medicine and IRCAD, in addition to perform collaborative work with almost 200 laboratories worldwide. This allows ICube to be one of the main driving force for research in Strasbourg. Indeed, the various projects done at ICube lead to 50 patents and over 4,000 publications[1] (journals and conferences combined) since its creation. However, this does not prevent ICube from working closely with industry as a lot of contracts and partnerships have been developed with over 100 companies, from small and medium enterprises to international groups (Novartis, Cisco, General Electric...).

While the unifying theme of the laboratory is imaging, the fields of the 16 teams are quite diversed. These teams compose four different departments:

- Computer Science; working on both applied and fundamental research, rangin from compilation and proofs in geometry to next generation networks.
- Imaging, robotics, remote sensing and biomedical; working mainly on the development of new technologies for healthcare.
- Solid-state electronics, systems and photonics; working mainly on the development of nano-eletronic and photonic devices as well as renewable energies.
- Mechanics; working mainly on fluid mechanics, geothermal energies and civil engineering.

My internship took place in the Computer Science department, managed by Prof. Pierre Gançarski, and composed of six teams working on various topics:

- Geometry and Computer Graphics
- Scientific and Parallel Computing
- Complex Systems and Translational Bioinformatics
- IMages, leArning, Geometry and Statistics
- Data Science and Knowledge
- Networks

[1] https://icube-publis.unistra.fr/

The Networks Team, which hosted me during my internship, is managed by Prof. Thomas Noël. It is composed of eight Ph.D. student, one engineer and ten faculty members. The Network Research Team focuses mainly on two different research topics:

- The Internet of Things, mainly centered around the design and the study of new efficient protocols and algorithms for the communication between constrained smart devices.
- Core networks, focusing on the design and the study of algorithms, protocols and topologies used to exchange data over computer networks.

My internship was related to the latter. During six months, I worked with Dr. Pascal Mérindol on the design and formalization of **OPTIC** (Optimal Protection Technique for Intra-domain Convergence), a new routing architecture database, which aims at predicting BGP convergence upon failure.

## 2. Problem Statement

With the growing popularity of real-time media and various applications sensitive to delay and losses, network operators need to ensure increasingly stricter Quality of Service to comply with the agreed Service Level Agreements **(SLA)** . Therefore, Internet Service Providers **(ISPs)** require their networks to react quickly to events such as failures or reconfigurations, which often result in an internal logical topology change that may introduce additional latency or packet loss.

Being dynamic objects by nature, IP networks suffer from frequent topology changes. These changes, voluntary or not, modify the logical topology of the network and thus impact intra and inter-domain routing. Indeed, while intra and inter-domain routing protocols (which ensure connectivity within a network and between multiple networks respectively) are two distinct entities having different purposes, they are tightly coupled, and their interaction determines the path followed by the outgoing traffic. This dependence creates a complex dynamic between the two protocols which may, upon internal failures or changes, temporarily hinder the forwarding of the transiting traffic and result in long-lasting consequences, i.e., packet loss or SLA violations until routing protocols finds the new best route. This update may be quite long (up to several minutes), due to forwarding information base architectures that do not handle inter-protocols interactions efficiently and that may be unadapted for the amount of information that inter-domain routing protocols need to store.

While current solutions succeed at mitigating the loss of packets upon topology changes , they do not always guarantee protection against every possible failure, neither the optimality of the backup path they switch to. Consequently, transiting traffic may still suffer from losses upon some events, or go through a non-optimal route, potentially leading to congestions or SLA violation due to increased delay or lowered bandwidth. Besides, by switching to non-optimal routes, these solutions use temporary backup paths, meaning that the path is bound to change again later upon convergence, disrupting the associated flows.

The objective of this work is to clearly identify current flaws in the standard forwarding information base architecture and associated solutions, while designing a new technique whose main goal is to predict BGP convergence upon failure. This new architecture should mitigate the effect of intra and inter-domain routing dependence, and therefore allow to immediately switch to the new best path upon any internal failure while bypassing the long and tedious BGP decision process [18]. In the following chapters, we formalize and prove OPTIC, before analyzing its complexity.

## 3. Outline

This thesis is organized as follows. Chapter 2 reviews the different routing protocols, focusing on BGP, as well as the different iBGP architectures and the associated problems that may ensue, in order to introduce the networking context of the issues OPTIC aims to solve. Chapter 3 introduces our personalized formal model we will be using to describe existing technologies as well as OPTIC. We will show in detail the negative impact of BGP-IGP interactions and the issues of the current solutions. Our contributions start at Chapter 4, where we describe our method, OPTIC. We then formalize and prove the algorithms and data structures that will be used to ensure that we can predict the post-convergence route choice upon any internal failure. Finally, Chapter 5 concludes this thesis by summarizing the contributions and describing possible future works.

*A glossary (page 61) as well as the nomenclature and a simplified summary of the theorems, lemmas, properties and definitions (Appendix A) used are available at the end of this thesis.*

# Chapter 2 Context: Routing Protocols

When packets go through a network, we usually want them to follow the best possible path towards their destination. The process of selection of these best paths is called routing and is performed by devices called routers. Routers compute the best paths and store the information in two dedicated information bases: the Routing Information Base **(RIB)**, which contains all routing information, and the Forwarding Information Base **(FIB)** which contains only the necessary information to forward traffic. While it is possible to populate these bases manually, by performing static routing, routers usually exchange the necessary information through routing protocols to populate their FIB and RIB.

Routing protocols allow routers to exchange reachability information in order to learn routes towards destinations. Routes exchange routing information, which they use to select the best route. To do so, routers exchange *signatures*, which describe the paths of the underlying network. Each router shares its own view of the network with its neighbors, such that this information is known throughout the network.

Within the same network, reachability between machines is maintained thanks to *intra-domain* routing protocols. However, the Internet being composed of a multitude of networks, these latter have in turn to be interconnected with one another. The routing information necessary to maintain this interconnection is exchanged via *inter-domain* routing protocols. In both cases, upon the reception of new routing information, routing algorithms are run to find the best route, or path, towards each destination. Routing protocols rely on routing algebras [12] allowing them to perform operations on the set "$\Sigma$" of path signatures they exchanged. Namely, the two main operations needed are the ability to compare and extract the best element ($\oplus$) between two elements of $\Sigma$ (i.e., the paths' signatures) and to combine ($\otimes$) them (i.e, extend the path by adding an edge). In addition, the operators $\prec$ and $\succ$ are used to compare paths, i.e, $\alpha_1 \oplus \alpha_2 = \alpha_1$ if $\alpha_1 \prec \alpha_2$. The notion of best path may vary: it may be the path minimizing the distance between to nodes, the delay, the latency, or any other metric. Consequently, the operations lying behind $\oplus$, $\otimes$, $\prec$ and $\succ$ change depending on the protocol they describe.

The exchange of signatures is done through the *control-plane*, which is in charge of constructing a map of the network, while the handling of the traffic itself is done by the *data-plane*. The interaction between the two planes theoretically ensures optimal packet forwarding. While several different intra-domain routing protocols are used in the Internet, the de-facto standard inter-domain routing protocol is the Border Gateway Protocol (BGP).

## 1. Interior Gateway Protocols

Different intra-domain routing protocols exist and can be used to ensure the connectivity between machines within the same network. These protocols may be classified into two categories: distance-vector protocols and link-state protocols. To compute the best paths, routing protocols use an abstract representation of the network, where nodes represent routers and links represent the respective adjacencies. **IGPs** are fairly simple protocols, as their main goal is to minimize the sum of the *weight* of the links used to

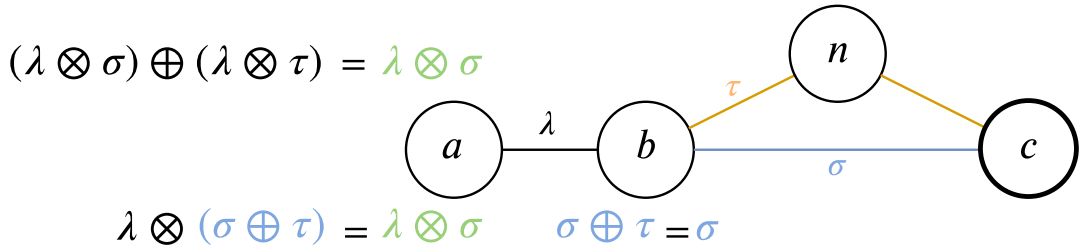



Figure 1: Distributivity of $\otimes$ over $\oplus$ in IGPs, with $\sigma \prec \tau$

reach a node. The weight of a link can be described as a scalar or vector which represent a metric, such as the strength of the connection, the cost of using the link, the bandwidth, delay and so on. These weights can be assigned automatically or statically. Therefore, when used to described IGPs, $\otimes$ can be seen as a simple addition (adding the weight of the new edge to the weight of the path), and $\oplus$ as a **min** operation (choosing the path with the minimal weight). Similarly, $\prec$ and $\succ$ are equivalent to the scalar comparison operator $<, >$, as we simply compare the weight of two paths.

IGPs rely on a simple monotone routing algebra. In particular, this algebra allows the distributivity of $\otimes$ (combination) over $\oplus$ (comparison):

$$\textbf{distributivity: } \forall(\sigma, \tau, \lambda) \in \Sigma^3 : \lambda \otimes (\sigma \oplus \tau) = (\lambda \otimes \sigma) \oplus (\lambda \otimes \tau)$$

This distributivity makes the result of $\oplus$ to be identical before and after any edge of the graph. In practice, this property allows to create a total global order between every routes across all routers. As all routing information is shared and coherent across devices, IGPs allow each node to use the global optimal route to a given destination: ever router can use its most preferred path. In this thesis, we will use $\alpha$ when referring to an intra-domain path. Since an intra-domain path is only characterized by its weight, $\alpha$ may equivalently refer to the path itself or its weight. So, for two distinct paths $\alpha_1$ and $\alpha_2$ to a given destination, such that $\alpha_1 \prec \alpha_2$,the result $\alpha_1 \oplus \alpha_2 = \alpha_1$ will be the same at any given node of the network.

This property is illustrated in Figure 1: computing the path choice directly at $a$ would give the same result as performing $b$'s choice first, then adding $\lambda$: $a$ does not have to take $b$'s choice into account when computing its best path towards $c$. Consequently, if $a$'s best path towards $c$ goes through $b$ and $\sigma$, then $b$'s best path to $c$ has to go through $\sigma$ as well. This global coherence induced by the distributivity property allows IGPs to be fairly resilient to convergence problems.

Despite routing protocol usually relying on a weighted graph to perform the computation of best paths, this does not mean that all nodes have a complete view of their network. For example, distance-vector protocols rely solely on the knowledge of the distances between one router and the others, while link-state protocols allow routers to create a more or less complete logical map of the network.

## 1.1 Distance-vector protocols

Nodes running distance-vector protocols know of how far they are from a given destination by maintaining an array of distances (usually the number of hops) from themselves to each destination. Thus, nodes possess a local view of the network, which only allows them to know which neighbor is the closest to another node, thanks to the associated distance. This array is sent to immediate neighbors periodically and when a failure occurs, in order for every node in the network to maintain an up-to-date database of these

distances. Distance-vector protocols, such as the well-known Routing Information Protocol [20] **(RIP)**, usually compute the best path to each destination using a distributed variant of the Belmann-Ford algorithm.

Distance-vector protocols suffer from a number of issues. Namely, the Belmann-Ford algorithm does not prevent routing loops. This family of protocols is also subject to high convergence time when used on large networks. In addition, simple versions of distance-vector protocols chose their route solely based on the number of hops without taking into account latency or other metrics reflecting the actual quality of a given path, meaning that for two given paths $\alpha_1$ and $\alpha_2$, $\alpha_1 \prec \alpha_2$ if the number of hops in $\alpha_1$ is inferior to the number of hops in $\alpha_2$. Finally, the fact that each node only possesses a local view of its network creates some issues, such as the *count-to-infinity* problem. In practice, these protocols are depreciated and the use of link-state routing protocols is vastly preferred.

## 1.2 Link-state protocols

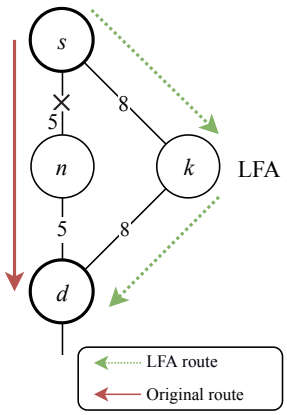
Nodes running a link-state protocol flood the states of their links not only to their immediate neighbors, but through the entire network. This flooding allows nodes to construct and maintain a map of the network in the form of a weighted graph. Routers running link-state protocols use the Dijkstra algorithm on the graph they constructed to compute the shortest path to each destination and populate their routing table. The weight of the edges may represent different metrics, such as latency or bandwidth, and is used to compare the different edges when computing the best paths. In such protocols, $\alpha_1 \prec \alpha_2 = \alpha_1$ if the sum of the weight of the links composing $\alpha_1$ is inferior to the sum of the weight of the links composing $\alpha_2$. The best-known link-state protocols are Open Shortest Path First [21] **(OSPF)** and Intermediate System to Intermediate System [23] (IS-IS).

While link-state protocols require more memory and processor power than distance-vector protocols, these protocols scale better and, thanks to the routers' global knowledge, are less susceptible to routing loops. However, since nodes might process updates at different times, their view of the network might differ. As a result, forwarding loops may occur until all nodes fully process their updates and share the same logical representation of the network again. Consequently, the correctness of such protocols relies on the fact that each node share the same coherent view of the network.

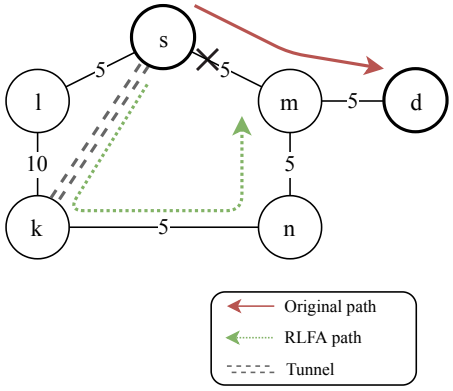
Both families of protocols belong, as stated above, to the intra-domain routing protocols. They allow a single network to achieve connectivity between every of its node. Inter-domain protocols, on the other hand, are used to interconnect these networks.

## 1.3 Fast ReRouting

Although standard intra-domain routing protocols tend to converge quickly [8], the emergence of low latency requirements for real-time applications, such as Voice over IP, translated into a need for even faster re-routing upon failure. A failure, once detected, triggers a reaction within the control plane and the re-computation of the shortest paths. Together, the detection and re-computation may create a delay too long for some sensitive applications during which packets are dropped . Some methods to mitigate such effects involve re-using still up-to-date knowledge of the network to perform only a partial re-computation of the best paths. However, most techniques, and in particular the ones that are going to be explained in this section, rely on pre-computed back up paths. These methods, referred to as Fast ReRoute mechanisms **(FRR)** allow to achieve a faster recovery time.




(a) Loop Free Alternate example




(b) Remote Loop Free Alternate example

Figure 2: LFA and RLFA illustrated. Node $s$, in figure 2a, uses a directly connected LFA to each $d$, while it uses $k$'s LFA ($n$) in figure 2b as no direct neighbors satisfy the LFA inequalities.

MPLS FRR mechanisms [24] are based on providing a backup tunnel either to create a new end-to-end path or to bypass a link or node which failed. To be able to react quickly, backup paths usually originate from the node immediately adjacent to the failure (i.e., the first router being aware of the failure) and merges back with the primary path later downstream. Since this decision is taken locally by the node next to the failure, the time take by the node to react is around the tens of milliseconds [28]. Once the detour is set up, the first node of the tunnel may redirect the traffic through a different tunnel at a later time. Backup paths may be computed by each node to predict the failure of each directly connected neighbor, adjacent link, or both.

IPFRR [28] does not require the network to use a tunneling protocol in order to work. Equal Cost Multi-Path **(ECMP)** can be considered an IPFRR technique, as it allows traffic to use other paths of equal cost upon a failure. ECMP is an extension to IGPs that allows the simultaneous use of multiple best paths sharing the same distance towards a destination. It is also widely used for load balancing. The best-known IP protection technique are loop-free alternates **(LFAs)** [1], which aim at delivering loop-free rerouting alternative path by using a pre-computed alternate next-hop when needed. A loop-free alternate of a router $s$ towards a destination $d$ can be described as a direct neighbor $k$ whose best path towards $d$ does not go back through $s$, such that no loops arise when the traffic is rerouted. Figure 2a gives an example of an LFA. A loop-free alternate can be formally described as a neighbor that satisfies the following condition: $cost(k, d) < cost(k, s) + cost(s, d)$, which ensures that no shortest path from the alternate node $k$ to $d$ goes through the failed component $s$. For example, in Figure 2a, we can see that $k$ verifies $8 < 8 + 10$, and is thus an LFA for $s$ towards $d$.

While ECMP and LFA allow to protect about 80% of failure cases in real-world IGP networks [28], some topologies may not allow for LFAs to exist. However, other mechanisms, such as Uturn Alternates may be used instead. U-turn Alternates aims at extending the protection coverage by looking for LFAs one hop further. Thus, a node $s$ uses $k$ as an Uturn alternate if $k$ has itself an LFA protecting $s$ for a destination $d$. Note that since $s$ does not have any LFA, $k$ does not satisfy the equation presented above and uses $s$ as a primary next-hop towards $d$. Consequently, a mechanism should be implemented for $k$ to recognize the traffic that needs to be sent to $s$' LFA, either by modifying the packets' header or by detecting rerouted traffic. Uturn alternates thus allow routers to bypass failed link by using LFAs that are not directly connected neighbors, but one hop further.

The range of the LFA mechanisms may be extended even further thanks to Remote LFA [3] **(RLFA)**. RFLA extends LFAs by allowing a router to find a LFA which is not directly connected, by setting up a virtual link to this remote LFA via a tunnel. The remote LFA, called a staging point, is selected so that, with no additional failure, the traffic will travel to the destination via normal forwarding paths without looping back.

Figure 2b shows an example of a remote loop-free alternate. Indeed, we can see that the standard path used by $k$ to reach $d$ does not go through $s$, meaning that $k$ is a suitable RLFA for $s$ to $d$. These LFA can be reached through IP-in-IP [29] tunnels or MPLS tunnels if available. Remote LFA allows LFA principles to work in topologies where some nodes may not have any neighbor that can be elected as LFAs. While some other FRR mechanisms may be used, such as Not-via Address [4], FIFR [22], or MRT-FRR [5], the most widely used techniques remain RLFA, LFA and MPLS FRR. While RLFA may protect an entire network when the links' weight are symmetric, it is not the case with asymmetric weights. However, recent solutions that use *Segment Routing* [7] allow to provide complete protection from internal failures by establishing repair paths easily, whatever the topology [19].

These solutions are able to reduce the recovery time upon failure within the domain by reacting more quickly that the IGP control plane. However, internal failures may have an effect at a larger scale and impact the inter-domain routing protocols, as the two are often tightly coupled. These types of failures are often the cause of long-lasting connectivity loss.

## 2. Border Gateway Protocol

The Border Gateway Protocol [25] **(BGP)** is the standard de-facto inter-domain routing protocol. BGP, being a *path vector* protocol, does not fall in either of the categories described in section 1. Path vector protocols rely on the path information itself to chose the best path. Indeed, using standard metric does not make sense for inter-domain routing, as going through some domain may be economically more interesting than going through others, creating the need to study the whole path to make a routing decision. Moreover, path vector protocols guarantee loop-free paths. Indeed, as the path taken by advertisements is accumulated at each node and carried within the message, making it easy for a node to prevent routing loops by checking of it already appears within the path description. BGP does not completely satisfy the criteria of path-vector, particularly since routing information may be hidden to neighboring AS due to economical relationships but is still usually considered within this family of protocols. While some alternatives such as EGP or EIGRP exist, they are either obsolete or proprietary and thus not widely used. BGP allows different routing domains, known as Autonomous Systems **(AS)**, to exchange routes between one another. Currently, the Internet counts with more than 60000 AS [30].

In practice, AS are identified by an Autonomous System Number **(ASN)**. Border routers **(ASBR)** of an AS exchange information with the ASBRs of the neighboring AS. Routers that run BGP are referred to as *BGP speakers*. When exchanging the route towards a given prefix, an ABSR adds the ASN of its AS to the `as-path` attribute of the advertisement. In addition to preventing loops, the `as-path` is one of the several path attributes taken into account when choosing the best path towards a destination. In turn, the routes learned by an ASBR from a neighboring AS also have to be advertised to the BGP-aware routers within the same AS, to ensure that each ASBR from the same AS knows routes for all destination. Information exchange between AS is assured by external BGP **(eBGP)**. More precisely, eBGP allows to send the best route for a

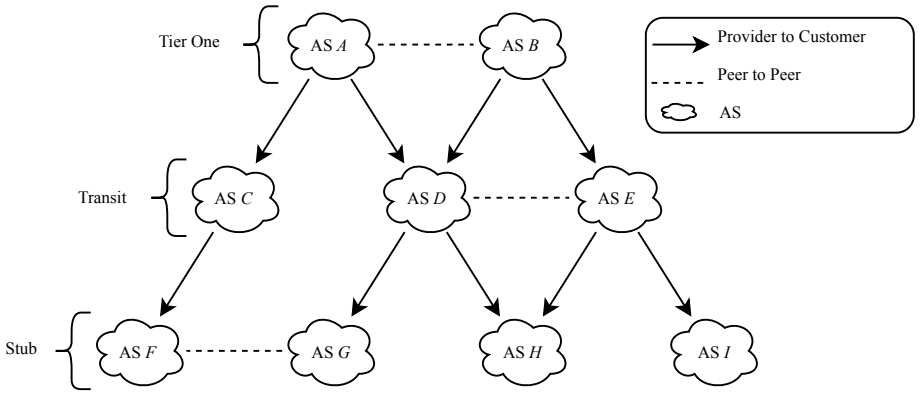



Figure 3: hierarchical relationships between AS

destination to the neighboring AS. In turn, a BGP router will send its best route, and only its best route, to each destination to its peers from the same AS via internal BGP **(iBGP)**. Since a router does not advertise a route learned via iBGP to another iBGP peer, iBGP routers are usually interconnected through a full-mesh to ensure the proper propagation, or *dissemination*, of routing information within the AS. The iBGP logical topology does not need to follow the physical one.

AS can be classified into several classes:

- Stub AS exclusively forward traffic originated from or destined to themselves;
- Transit AS may forward traffic from their client, but are themselves clients of other AS;
- Tier-1 AS do not purchase transit service from any other AS.

To achieve full connectivity, AS may purchase transit service from a provider, or establish peering relationships with neighboring AS, allowing a theoretically free exchange of traffic between both AS In BGP, economical relationships are a key factor influencing the best route decision process.

## 2.1 AS relationships

While the best path is globally imposed by the weights of the link of the logical topology when using a link-state protocol, the best BGP route is elected according to a collection of characteristics which can be locally created and exchanged by each node. Indeed, BGP extends the scalar used to compare paths in IGPs to a vector by adding a list $\beta$ of attributes. These attributes, are in turn used to choose the best route towards a destination. Therefore, Path vector protocols such as BGP allow operators to create their own exchange policy, making them very flexible and especially suited for inter-domain routing where economical relationships create complex dynamics.

Indeed, these relations creates a hierarchical structure within the Internet. An AS can have peers, customers and providers. Stubs AS may also be clients of transit AS which may be clients of Tier-1 AS. These *customer-provider* relationships are to be distinguished from *peer-to-peer* relationships, which link AS having usually comparable sizes. While customer-provider relationships involve a pricing model where the customer pays a bigger AS to access the Internet, it is mutually advantageous for peers to exchange each other's traffic. This hierarchical organization is illustrated in Figure 3. The exchange of routes done between each AS depends on their theoretically follows the Gao-Rexford policies [9]:

- An AS should export its routes and the routes of its customers to providers, but should not export the routes learned from a peer or another provider

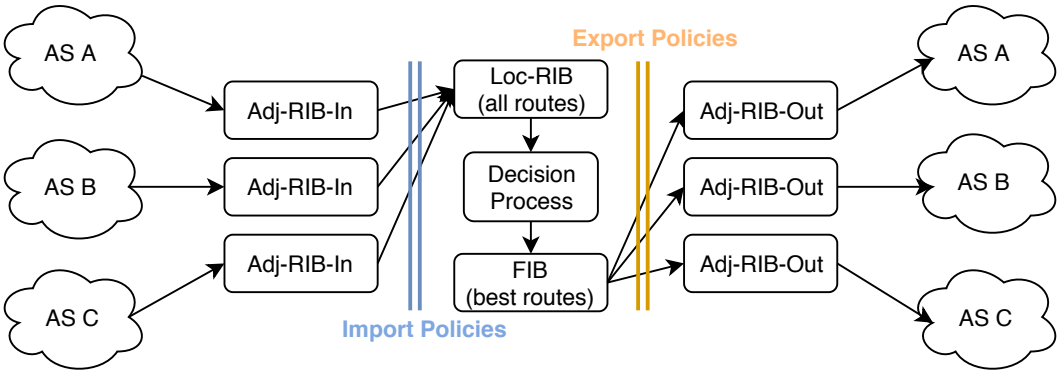



Figure 4: Routing information base of a BGP router

– An AS should export all of its routes to its customers or siblings.
– An AS should prefer the route learned from its clients over the routes learned from a peer, and should choose the routes learned from its providers if no other options are left.

These rules translate what is called the *valley-free* property: after going through a customer-to-provider or peer-to-peer link, the traffic should not go through a customer-to-provider or peer-to-peer link again. In other words, once the traffic has gone up one level in the hierarchy, it may either stay at the same level for one step only or go back down, hence the name. These rules makes sense economically: in Figure 3, if AS $D$ export routes learned from its provider AS $A$ to its peer AS $E$, AS $E$ might use this route, making AS $D$ pay for the traffic. However, there is no obligation for an AS to follow these rules.

The announcement of routes to other BGP speakers are done through several different type of messages.

– `OPEN` messages are sent to establish the connection between two BGP speakers;
– `KEEPALIVE` messages are sent to ensure that the BGP speaker is still reachable;
– `NOTIFICATION` messages are sent to notify another BGP speaker of an error;
– `UPDATE` messages are used to exchange information with another AS. `UPDATE` messages are also used to `WITHDRAW` routes.

To save bandwidth and processing power, BGP uses incremental updates. Once complete routing information is sent during the initial exchange, BGP speakers exchange only the changes made to that information. The `UPDATE` messages contain the destination prefix, along with attributes bound to the route towards these prefix. As mentioned earlier, these attributes are used to select the best possible routes.

## 2.2 BGP route selection

BGP speakers store all of their routing information in their Routing Information Base (RIB). The way information is stored and exported is illustrated in Figure 4. Routes learned from other AS are stored in the *Adj-RIB-In*. These routes are then filtered according to import policies and only *valid* routes are kept. Import policies may, for example, dictate to reject prefixes above a certain length. Routes that were not filtered out are stored in the Loc-RIB. Adj-RIB-Outs contain the routes that will be sent to BGP peers, and which were not filtered out by export policies. There is one Adj-RIB-In and one Adj-RIB-Out per eBGP session, as the exported and imported routes and policies might differ depending on the AS it originated from or the AS they will be sent to.

| *Step* | *Criterion* | |
|---|---|---|
| 1 | Highest `local-pref` LP | (economical relationships) |
| 2 | Shortest `as-path` AS | |
| 3 | Lowest `origin` | (IGP or EGP over `unknown`) |
| 4 | Lowest MED | (cold potato routing) |
| 5 | Prefer routes learned via eBGP over iBGP | |
| 6 | Lowest IGP cost I | (hot potato routing) |
| 7 | Lowest `router-id` rid | (arbitrary tie-break) |

Figure 5: Steps of the BGP decision process

As an AS can receive multiple routes to the same destination that respect the import policies (for example, routes originated from different AS), the Loc-RIB might contain several routes towards the same prefix. The BGP attributes are then used to rank the routes. The way routes are usually ranked is illustrated in Figure 5. Each criterion comes into play whenever paths could not be differentiated thanks to the previous one. The comparison between BGP paths can thus be seen as a lexicographical comparison of the paths' BGP attributes. Therefore, the $\oplus$, $\prec_I$ and $\succ_I$ operator can here be seen as a **min** operation applied on vectors lexicographically, and lexicographical comparison operators. However, the extension ($\otimes$) done by BGP is quite more complex than its IGP counterpart. Indeed, paths may be filtered out or have their attributes changed upon extension to reflect policies and local preference. Consequently, the result of $r^1 \otimes r^2$ when extending BGP routes depends on more factor than the routes' characteristics, such as local policies or preferences which allow to rank the extended new route at any given position and thus influence the selection of the best route.

This route selection is referred to as the BGP *decision process*. In practice, the last attributes are here to ensure that no two routes can be ranked equal at the end of the decision process. We refer to such kind of attributes as `Tie-Break`. In this report, we represent BGP routes as an triplet containing a destination $r_a$, a gateway $r_g$ and, in particular, a vector of attributes $r_a$, defined as

$$r_a = (r_a[0], ..., r_a[7]) =$$
$$(\texttt{local-pref}, \texttt{as-path}, \texttt{origin}, MED, \texttt{eBGP}, \texttt{IGP-cost}, \texttt{Tie-Break})$$

For readability purposes, we use the following notation:

$$r_a = (r_a[LP], r_a[AS], r_a[O], r_a[M], r_a[EBGP], r_a[I], r_a[TB])$$

The $\prec$ and $\succ$ operators we used in earlier sections to compare paths are equivalent, when used to compare *valid* BGP routes , to lexicographical comparison operators. Thusly, for two routes $r'$ and $r''$ characterized by their vectors of attributes $r'_a = (r'_a[LP], r'_a[AS], ..., r'_a[TB])$ and $r''_a = (r''_a[LP], r''_a[AS], ..., r''_a[TB])$ respectively, $r'_a \prec r''_a$ if, with $i$ the first index where $r'_a[i]$ and $r''_a[i]$ differ, $r'_a[i] < r''_a[i]$. Simply put, $\prec$ allows us to compare BGP routes the same way the BGP decision process does. For example, for two routes $r'_a = (1, 1, 1, 2, 1, 10, 1)$ and $r''_a = (1, 1, 2, 1, 1, 3, 4)$, $r'_a \prec r''_a$.

The `local-pref` attribute, being the highest on the list, is used to override the rest of the BGP decision process. It may be used to enforce economical relationships, for example, to prefer routes learned from clients. The `as-path` attribute is used to choose the route going through the less number of AS. The Multi-Exit Discriminator (MED) is set by the external BGP peer to indicate the preferred point of entry into its AS. The MED is only used to compare routes that originate from the same AS. As a result,

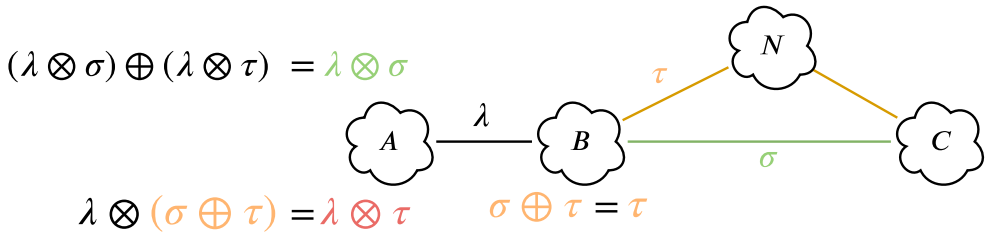



Figure 6: Loss of Distributivity of $\otimes$ over $\oplus$ in BGP

the MED may break the total order between BGP routes and lead to anomalies such as MED oscillation [26]. The IGP metric allows to prefer the route having the egress router (i.e., the *gateway* through which the traffic will leave the AS) with the lowest IGP cost. This rule is referred to as *hot-potato routing*, as its main point is to pass the traffic onto another AS as quickly as possible.

While some attributes such as local-pref and as-path are related only to inter-domain routing, some result from intra-domain routing, such as the IGP distance towards the gateway. Consequently, we can see the vector of attribute describing a BGP route as a composition $r_a = \beta \circ \alpha$, with $\beta$ representing inter-domain routing related attributes and $\alpha$ an IGP-related attribute $\alpha$. Although $\beta$ does not change across the routers being aware of the route within the same AS, $\alpha$ changes depending on the IGP distance of the router processing the route from the advertising gateway.

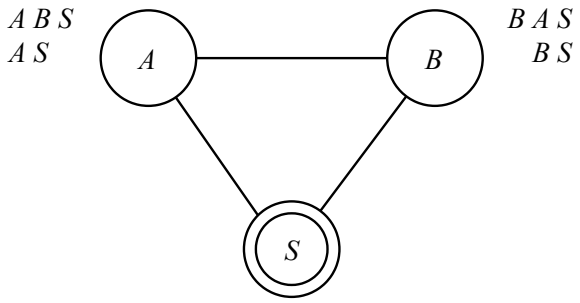
Once the best paths are selected, their associated next hops are resolved and the outgoing interface that should be used to reach them is computed. The next hop information and associated interface identifier is stored in the Forwarding Information Base (FIB). The best route for each prefix is then retransmitted to BGP peers. The complex route ranking process and the fact that incoming and outgoing advertisements may be filtered out adds a layer of complexity to BGP. In particular, the distributivity of $\otimes$ over $\oplus$ evoked in section 1 does not necessarily hold true in BGP.

While IGPs manage to make each node reach a global optimal decision for each route, BGP allows each node to chose a local optimal path according to their own ranking and routing information, which may differ from their neighbors'. In other words, while a local optimal path is part of a global optimal path in IGPs, this is not true in BGP as the notion of global optimal path is lost. As can be seen on Figure 6, the loss of the distributivity property is due to the fact that there is no global ranking of BGP routes, in addition to the fact that route information may be hidden to other nodes. For example, as $B$ prefers $\tau$, $A$ will have to chose $\tau$ as it is the only path that $B$ will advertise, even though $A$ would prefer $\lambda \otimes \sigma$.

This loss of global coherence may also be equivalently described by the loss of the *isotonicity* property [31] using the aforementioned $\prec$ operator:

$$\textbf{isotonicity: } \forall(\sigma, \tau, \lambda) \in \Sigma^3 : \sigma \prec \tau \implies (\lambda \otimes \sigma) \prec (\lambda \otimes \tau)$$

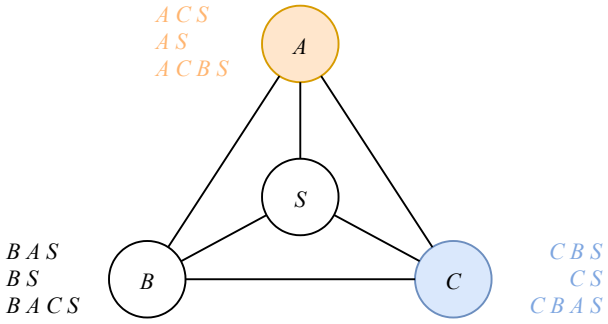
Isotonicity implies that a node cannot make the opposite choice of its neighbor when comparing two routes. With an non-isotonic algebra, it is possible that $\tau \prec \sigma$ but $(\lambda \otimes \sigma) \prec (\lambda \otimes \tau)$ hold true. For example, $(\lambda \otimes \tau)$ might even be forbidden for some nodes because of policies. Consequently, the preferred paths of some nodes may be filtered out by their neighbors. The fact that nodes chose and filter routes following an egocentric behavior induces the loss of global coherence, leading so to potential convergence problems within BGP.




(a) DISAGREE GADGET

| $i$ | *Activation* | *Node A* | *Node B* |
|---|---|---|---|
| 1 | $\{(S,A),(S,B)\}$ | $(A\ S)$ | $(B\ S)$ |
| 2 | $\{(A,B),(B,A)\}$ | $(A\ B\ S)$ | $(B\ A\ S)$ |
| 3 | $\{(A,B),(B,A)\}$ | $(A\ S)$ | $(B\ S)$ |

(b) Infinite activation sequence




$$ACS \prec ACBS \equiv (AC \otimes CS) \prec (AC \otimes CBS)$$

$$CS \succ CBS$$

non-isotone

(c) BAD GADGET

Figure 7: Two gadgets showing BGP convergence issues

This was formalized by Griffin et al. [11]. The formalization proposed models that allow to represent convergence problem in BGP as an undirected graph, whose nodes represent the different AS, and the links the peering BGP relationships between them. Each AS has a list of routes to the AS $S$ ranked in order of preference. Iteratively, a subset of AS applies its import rules, chooses its best available route accordingly, and exports it to other AS. This process is executed during the *activation* of the subset of nodes. The subset of nodes being activated might replace their current path with a higher-ranked one, or a lower-ranked one if the activation of another node made them lose their current preferred path. The system represented may follow two different behaviors. The system may converge, in which case routers cannot improve their routes no matter the subset of nodes being activated. In this case, the system is said to have a *stable state*: each node found the best possible choice within its options for the given scenario. The system might also diverge, meaning that routers will keep changing their best routes forever. Figure 7a shows an example of the latter case, called the DISAGREE GADGET.

The DISAGREE GADGET is especially interesting as it possesses two stable states. Either node $A$ chose the direct route to $S$, and $B$ will thus have to go through $A$ to go to $S$, or the other way around, depending on the activation sequence. These solutions assume that the two nodes are activated at different times, such that one of them has to choose its least preferred path. However, while two stable states exist for DISAGREE, it is possible for this gadget not to converge. Figure 7b shows the activation sequence leading to this unstable state. In this case, both nodes $A$ and $B$ chose their least preferred route at the same time, and exchange it with one another. Upon learning the choice of their neighbor, the nodes change their path in order to use their most preferred one, i.e. the indirect path towards $S$. Doing so, they prevent their neighbor to chose its most preferred path, leading them to change their direct path towards $S$ once again. This oscillation might continue indefinitely.

While DISAGREE can converge (in fact, as nodes are usually not perfectly synchronized in practice, a DISAGREE-like scenario would have a high probability of reaching a stable state), some gadgets do not possess any stable solution. Figure 7c shows such an example, called BAD GADGET. In this gadget, each node $A$, $B$ and $C$ prefers the route going through their clockwise neighbor. The first set of nodes being activated will choose

the direct path towards $S$, being the only path available to them originally. By doing so, these nodes allow their counter-clockwise neighbor to go through them to reach the central node. The node being allowed to chose this indirect path will prevent its neighbor to use it as a transit AS forcing it to chose the direct path. We can see that this cycle will repeat forever, as each time a new node will be forced to chose the direct path.

Note that we can see that these convergence problems are due to the loss of the isotonicity property. Indeed, the preference of each node in this bad gadget creates a non-isotonic routing algebra, as shown explicitly by the equalities in Figure 7c. While $C$ prefers the path $CBS$ over $CS$, $A$ prefers $ACS$ over $ACBS$: $A$ makes the opposite choice of its neighbor $C$ after the local processing of the routes. This can also be shown with `DISAGREE`. Indeed, in this gadget, illustrated by Figure 7a, both $BS \succ BAS$ and $ABS \prec ABAS \equiv (AB \otimes BS) \prec (AB \otimes BAS)$ hold true, making the associated algebra non-isotone. While the inequality $ABS \prec ABAS$ is not explicitly stated in the gadget, any route that is not present in the lists is filtered out and thus considered worse that any route that is in the lists. In the `DISAGREE` case, these filters manage to prevent dataplane loops, as routes that go back to the origin node (such as $ABAS$ in our equalities or $BABS$) are filtered out.

While there exist several instances of the sort highlighting BGP convergence problems, it has been shown that BGP converge if the guidelines explained in section 2.1 are followed [9]. More precisely, a BGP converges on a given topology if no *dispute wheel* [10] (a structure representing a circular set of dependencies between routing policies) can be extracted from the associated SPP. Policies that are Gao-Rexford compliant do not form any dispute wheel. When applied thanks to filters and preferences, these rules create an isotone routing algebra. However, convergence problems might happen within the same AS, if complex iBGP topologies are used.

## 2.3 iBGP architectures

Since economical relationships have no reason to exist within an AS, iBGP networks do not necessarily need to follow a hierarchical structure. However, it must be noted that iBGP routers do not re-advertise routes learned via iBGP peers to other iBGP peers. This simple rule allows to prevent routing information loops between iBGP peers. As a result, by default, iBGP requires a full-mesh between iBGP speakers in order to allow information to be distributed across all peers. This requirement makes the configuration of an iBGP network complex, and forces each router to maintain $\frac{n(n-1)}{2}$ sessions. As the size of networks grew along with the amount of routing information being exchanged, the need for a new way to distribute routes among BGP peer leveraging this full-mesh condition arose.

One of such solution was BGP confederations [33]. BGP confederations are used to subdivide an AS. These sub-ASes have to be fully meshed. However, the number of peerings between sub-AS themselves has no such condition, reducing so the number of needed iBGP sessions. Nevertheless, the configuration process of BGP confederations is complex, making them far less popular than their more simple counterpart, *route reflectors*. Route reflectors [2] are a new kind of BGP speaker. Contrarily to standard BGP routers, route reflectors can re-distribute routes learned from iBGP peers. Thus, BGP routers may only peer with this central point, which centralizes and re-distributes routes learned from other BGP routers. Route reflectors thus allow to drastically reduce the number of needed iBGP session.

Introducing only one route reflector within an AS creates a single point of failure, as this kind of BGP speaker has a central role in the route distribution. To prevent

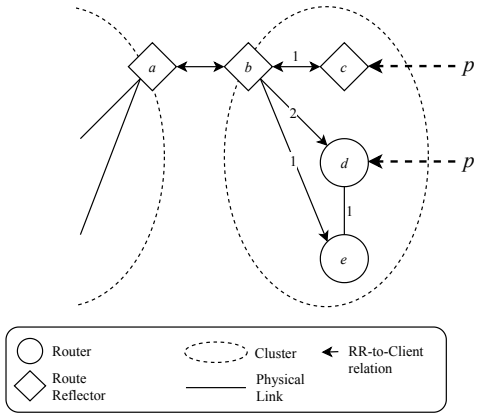



Figure 8: Route-reflector induced problems. The route reflector $a$ does not receive routes towards $p$ as $b$ its peer. $e$ would have a better path to $p$ through $d$ if the BGP speakers were fully meshed.

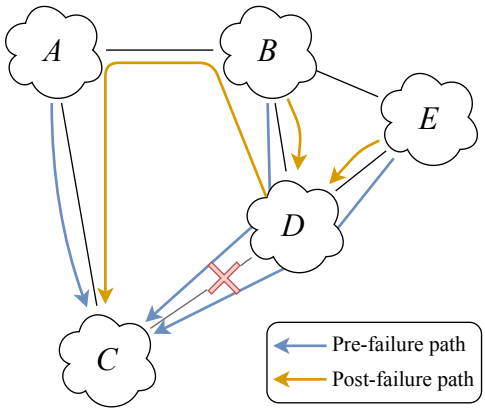



Figure 9: RBGP reacting to a link failure between $D$ and $C$. B announces a fail-over path towards C to D. When D $\rightarrow$ C fails, D uses the fail-over path. When B receives its traffic back, it knows that $C$ is now unreachable through $D$ and uses $A$ instead.

critical failures, several route reflectors can co-exist within the same AS. As well as providing redundancy, having several route reflectors within a single network allows to create a hierarchical iBGP topology. In such a kind of network, route reflectors have two kinds of peers: clients (e.g., a standard BGP speaker), and non-clients (e.g., another route reflector). Upon the reception of a route from a non-client peer, a route reflector redistributes the said route to all of its clients. If the route is received from a client, it is redistributed to clients and non-clients. A route reflector and its clients form a *cluster*. As do standard iBGP routers, route reflectors only forwards their best routes to their clients. While route reflection is a way to solve the scalability problem of iBGP, there are some caveats to these methods. In particular route reflection may also reduce path diversity, making routers adopt sub-optimal routes due to incomplete FIBs( [36], [34]) or induce forwarding loops [27] and route oscillation. As stated above, the chosen route might not be optimal for clients within the cluster, as it depends on the logical position of the route reflector. Indeed, for a given vector of attribute $\beta \circ \alpha$, as $\alpha$ might defer from router to router, a client of the cluster might not have chosen the same route as the route reflector of its cluster. While some solutions, such as BGP free cores thanks to tunneling techniques or BGP route propagation rules [13] mitigate some anomalies, they remain a possibility in most standard iBGP topologies.

Figure 8 illustrates both of these problems. Let us consider the rightmost cluster. Both $c$ and $d$ receive an advertisement for the prefix $p$. The router $b$ receives this route from $c$, a peer route reflector, and from $d$, a client. From the point of view of $b$, assuming all other attributes are equal, the route learned from $c$ is better, as the IGP cost towards the egress point is lower than the IGP cost towards $d$, leading to $(r^c = \beta \circ 1) \prec (r^d = \beta \circ 2)$ . Thusly, $b$ will re-advertise $r^c$, and $r^c$ only, to its clients. Due to this behavior, $e$, which only received the route towards $p$ from its route reflector $b$, will go through $c$ to reach $p$, while the node $e$ would have chosen the route learned from $d$ had the iBGP network been fully-meshed. Indeed, if $e$ had received an advertisement from $d$ towards $p$, the advertised route would have been equal to $r^d = \beta \circ 1$, while the route advertised by $c$ would have been equal to $r^c = \beta \circ 2$ from $e$'s IGP point of view. In addition, since $b$ learned this route from a route reflector, it will not re-advertise it to other route reflectors. Consequently, $a$ will not be able to reach $p$.

This route visibility problem can be mitigated thanks to BGP-Addpath [35]. BGP-Addpath is a solution which aims at preventing what is referred to as implicit withdrawals: an advertisement of a route with better attributes that replaces the previous route. BGP-Addpath allows to advertise multiple paths for the same destination. This, in turn, allows faster re-convergence, as an alternative path is available upon failure (though this path might neither be the best one nor still be reachable). Different policies are available in BGP-Addpath, allowing for example to advertise all the known paths towards a destination, or the N best paths. In addition to reducing convergence time, BGP-Addpath also allows to reduce the chances of suboptimal routing, as routers can make their own best choices instead of having to submit to their peers' choices. It also allows to reduce the chances of anomalies related to local preferences.

Another way of reducing the complexity of an iBGP network while still allowing the distribution of routes within the AS is to use *route servers*. Route servers [16] store all the route information available and forward it to all the nodes in its cluster. By forwarding the update as-is, without deciding of route suitability for its clients, a route server allows to maintain path optimality within its cluster (according to the BGP decision process), at the cost of heavy update exchanges.

## 2.4 BGP Fast ReRouting

As mentioned in section 1.3, the usage of applications requiring low latency created the need for fast rerouting techniques allowing to restore connectivity in tens of milliseconds. However, while these techniques were focused on restoring connectivity after an internal failure, such problems also occur at the inter-domain scale. While BGP may suffer from slow reaction time due to its FIB update time, other factors slow down the convergence of BGP. Namely, the control plane messages have first to be processed and propagated from node to node, and are often delayed by timers to prevent overreactions. Such failures may be the cause of several minutes of packet loss [18]. Therefore, the convergence time of BGP may be divided into two delays: the propagation delay, and the update delay. The propagation delay is the result of the propagation time of BGP control plane messages. The update delay is the result of the re-computation of the paths once the failure was detected. Thus, the further the failure is, the longer an AS will have to wait before being aware of it, as it will have to wait for AS on the path to process and relay the information. The solutions in these sections aim to reduce both delays by checking for data-plane hints to anticipate control-plan message and relying on pre-computed backup paths to bypass the computation of new paths

Various methods exist to mitigate the effect of slow BGP convergence upon remote outages. These methods are usually based on either predicting an outage thanks to the data plane or using the BGP control plane to guess the full scope of a remote failure. While another solution could be to reduce BGP convergence time, the propagation delay of BGP makes it unlikely for convergence to be achieved in a time that is viable for real-time applications, even if it were to be improved.

R-BGP [17], for example, aims at protecting the data from the effect of outages while BGP converges. To do so, R-BGP is based around the pre-computation of fail-over paths, which are then advertised to the BGP peers. The main goal of R-BGP is to eliminate packet loss during BGP convergence, and allowing an ISP to protect itself and its customer even if no other ISP cooperates. Using R-BGP, an AS will advertise one fail-over path per destination to the next domain along the primary path to the said destination $d$. Normally, following the rules explained in section 2.1, an AS does not advertise a route towards $d$ to their next-hop towards $d$, as, by doing so, they might offer

a path to $d$ to their provider. However, here, the goal is to detect quickly an outage which renders the route unusable. When the client receives back packets destined to $d$ from the next-hop interface of their primary path towards $d$, it knows that its next-hop lost its route towards $d$. Thanks to this data-plane clue induced back the fail-over path advertisement, the client knows that it should traffic through the fail-over path it advertised. An AS may want to advertise a fail-over to its next-hop to ensure that the next-hop does not drop the traffic upon failure. As an example, Figure 9 shows how R-BGP allows to reduce the loss of connectivity upon a remote outage. Thanks to R-BGP, $B$ advertised a fail-over path $B\ A\ C$ to $D$. Usually, the traffic from $B$,$E$ towards $C$ goes through $D$. However, once the link failure between $D$ and $C$ occurs, the traffic is able to be directly redirected through $B$. Upon the reception of transit traffic to $C$, $B$ can infer the failure, and use its backup path to reach $C$ without suffering any losses.

Similarly, Blink [14] is another solution which uses data-plane clues to bypass the R-BGP control-plane convergence to react to remote R-BGP failures. This data-driven convergence allows to react a lot quicker to remote failures, thus reducing the induced packet loss. To do so, Blink monitors TCP flows to detect hint of remote failures. To reduce the number of flows to track, only the active flows destined towards the most popular destination prefixes are monitored. Monitored flows are periodically replaced on the fly if they become inactive. To detect remote failure, Blink takes advantage of TCP retransmissions. Indeed, upon a failure, when packets are lost, TCP will retransmit data packets that were not acknowledged. A unacknowledged packet sent at time $t$ will be retransmitted several times up until its acknowledgement. These retransmissions are delayed further and further in time, following an *exponential backoff*. For example, a packet sent at $t$ may be retransmitted at $t + 300ms$, $t + 600ms$, $t + 1400ms$. This mechanism creates *waves* of retransmissions. Blink detects those waves of retransmissions on several destinations to detect remote outages. Upon detection, Blink uses pre-computed backup paths to redirect the impacted flows. In practice, Blink sends flows to each backup path to check which path remains unaffected and will be able to restore connectivity.

As mentioned earlier, some methods, such as Swift [15], uses the BGP control plane to reduce the reaction time upon failure. To do so, Swift uses only a fraction of the BGP control messages received after a failure to predict the localization of the outage and the complete set of affected prefixes. Indeed, while the propagation of route withdrawals is slow, these messages may be sent one at a time, prolonging so the time needed to know the full extent of a remote failure. By predicting the set of prefixes affected by an outage, one can prevent a substantial amount of traffic loss. Swift feeds both explicit and implicit information carried by BGP control-plane messages to an inference algorithm, which in turn tries to localize the failure and to predict the probability of each prefix to be affected by the outage. The traffic destined to these prefixes is re-routed towards backup next hops which were pre-computed according to BGP information. Once BGP converges, the traffic falls back to the standard BGP route.

While these solutions indeed allow to potentially reduce the time during which connectivity is lost upon a failure, they are focused around the detection remote outages. However, because of the intricate interplay that takes place between BGP and IGP, intra-domain failures may very well impact transiting traffic.

# Chapter 3 External traffic re-routing

As mentioned in Chapter 2, while BGP and IGPs are different types of protocols with different goals, the path taken by a transiting packet is decided by the interplay that occurs between them. IGP metrics first influence the BGP decision process, as the IGP distance between the router and the gateway linked to the path is one of the attributes used in the said decision process. In addition, the path taken by the traffic to reach the selected BGP next-hop is controlled by the IGP. These dependencies make the forwarding of transiting traffic complex and may create undesired side effects as IGP changes can impact BGP routing. We have shown in previous sections that the convergence time of routing protocols plays an important role in today's networks. As IGP changes may impact BGP paths, the delays introduced by these modifications can be significant.

The solutions evoked until now were either focusing on IGP routing or on remote failures. However, the BGP decision may be re-run due to intra-domain failures, as a consequence of the dependencies between IGPs and BGP.

## 1. Model

In this section, we introduce our own personalized model and notations. We will first use this model to describe the current technologies and methods, before using it to describe our solution, OPTIC, and the associated definitions and properties.

### 1.1 Traffic forwarding

Upon the reception of data, routers need to resolve the next-hop for the given destination, i.e., to dfind the best next-hop for this destination. This resolution relies on several steps which are illustrated in Figure 11. The next-hop related to a packet is decided according to its destination. Once the destination is extracted, a longest prefix match is performed. By doing so, the router finds the most specific prefix in its base containing the IP address of the destination, ensuring the most precise forwarding possible. The prefixes are stored using a radix tree or a similar structure. This data structure is especially effective to perform this type of lookup. A binary search is performed against the tree structure storing the entries: at step $i$, the search branches one way or another depending on if the $i^{th}$ bit of the tested address is 0 or 1. The matched prefix may either be an internal

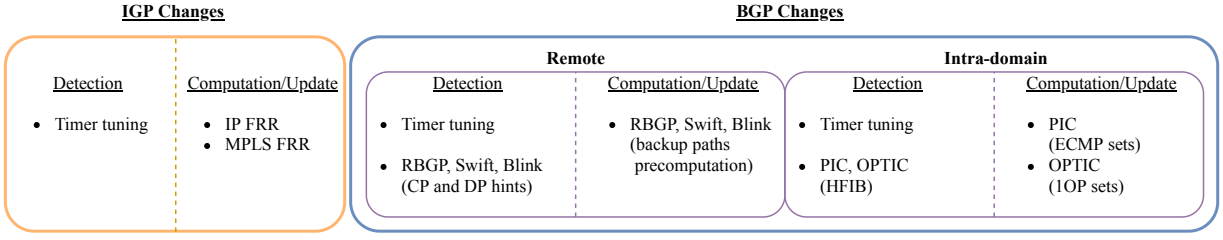



Figure 10: Methods that aims to mitigate the impact of failures on a network usually rely on decreasing the detection time or the computation and update time. The methods presented were focused on IGP and remote failures.

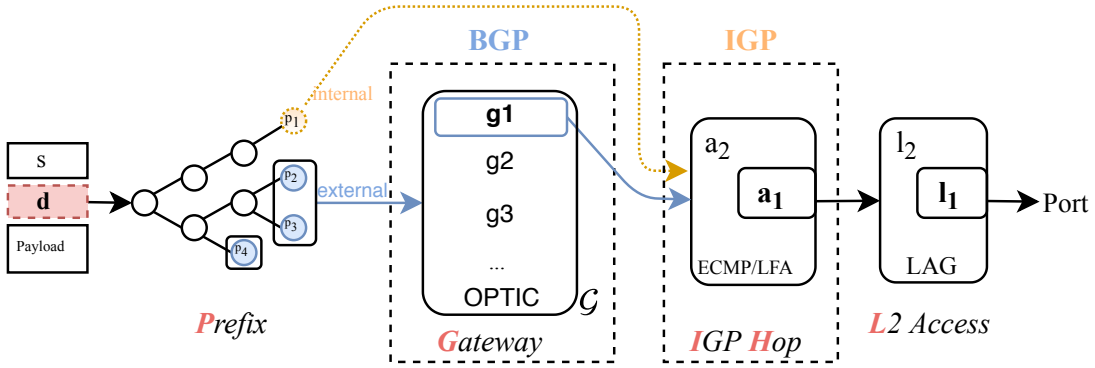



Figure 11: Forwarding Information Base and standard packet forwarding The destination prefix is extracted from the packet. The port the packet should be sent through is computed solely through the IGP for internal destinations, and through BGP and the IGP for external destinations.

prefix or an external prefix. If the prefix is internal, the interface that needs to be used is resolved only through the IGP. This resolution may be seen as the application of a function $IH$, which, given a prefix $p$, returns the corresponding IGP next-hop. Once the best IGP next-hop is found, the outgoing interface that should be used to reach $IH(p)$ can be resolved. This resolution can be seen as the application of the function $L$ which returns the best outgoing interface for an IGP destination $IH(p)$. While $IH$ and $L$ only return one element, this element may be chosen from a set containing equally valid values. This set is constructed thanks to routing strategies such as ECMP or LAG, which allow better resiliency upon an interface or IGP failure and to perform load-balacing. Thus, the resolution of the outgoing interface for a given packet towards an internal destination $d$ may be written as a composition of functions representing the various steps of the process described above:

$$Port(D) = L \circ IH \circ P(d)$$

However, this composition must be extended in order to be applied to external destinations. First, the router should find the best BGP next-hop (or gateway). This gateway is the best exit point to leave the AS when trying to reach $d$. The best BGP next-hop is computed via the BGP decision process detailed in section 2.2, represented here by the function $G$, which takes an external prefix and returns the best associated BGP **NH**. As the BGP NH is not usually directly connected, the IGP next-hop needed to reach it needs in turn to be resolved. the best IGP next-hop $a$ towards is thus computed, and in turn the associated outgoing interface. Consequently, the resolution of the outgoing interface for a given packet towards an external destination $d$ can be written as the following composition of functions:

$$Port(D) = L \circ IH \circ G \circ P(d)$$

## 1.2 BGP route ranking

The most simple implement of $G$ returns the best path towards $p$ by sorting all known routes (and associated gateways) towards $p$ via a lexicographical order performed on the attributes of each route. Each attribute is consider in the order of importance, and routes that do not possess the best value for a given attribute are filtered out. If needed, this selection ends on an arbitrary tie break to ensure that only one route can be selected. To simplify our model and subsequent explanations, only four attributes will be considered in the remainder of this report: `local-pref (LP)`, `as-path (AS)`, `IGP`

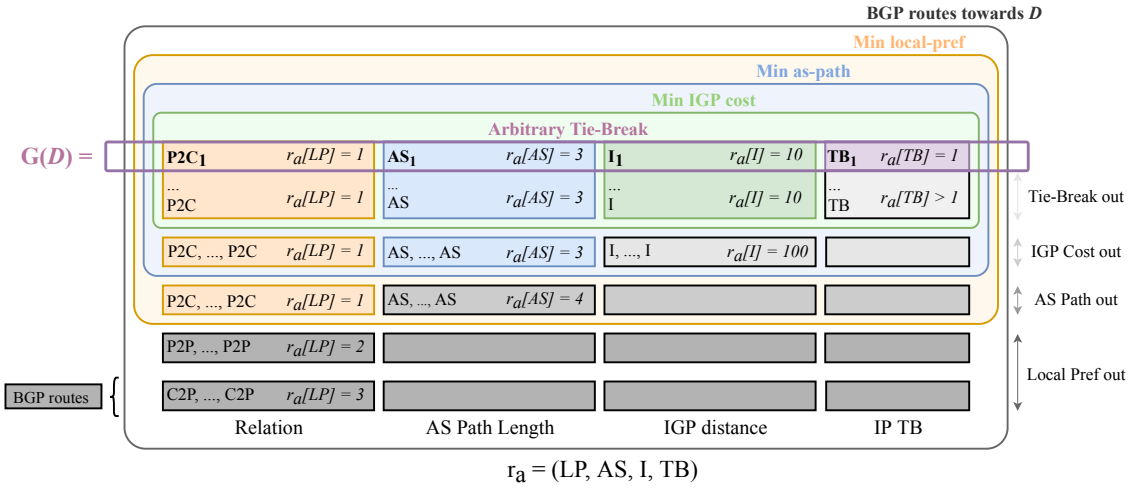



Figure 12: Graphical representation of the BGP decision process with our notations. Each attributes of each route is considered. Routes that do not possess the best value among all other routes for the given attribute are filtered out, until only one route remains.

`cost (I)` and `Tie-Break (TB)`. In particular, we slightly extend how we represent BGP routes. BGP routes $r$ will be considered as a triplet containing

- a destination $r_d$
- a vector of attributes $r_a$
- an associated gateway $r_g$

We simplify the vector of attributes of BGP routes to four field. BGP routes $r$ will be considered as having a vector of attributes $r_a$ equal to $(r_a[LP], r_a[AS], r_a[I], r_a[TB])$. This simplification is purely for readability purposes and does not induce any loss of generality. In addition, we extend the operator $\prec$, presented earlier in this report, which allowed us to compare routes by comparing their component lexicographically. We introduce $\prec_x$, which, as $\prec$, is a lexicographical comparison operator that can be applied to paths, particularly to BGP routes. However, $\prec_x$ stops the comparison right before the $x^{th}$ element of the vector. As an example, for two BGP routes $r'$ and $r''$ such that $r'_a = (1, 1, 1, 1)$ and $r''_a = (1, 1, 2, 2)$, then we have $r'_a =_I r''_a$, while $r'_a \prec r''_a$. In other words, if $r'_a =_I r''_a$, the two routes are tied in the BGP decision process up until the application of the `IGP cost (I)` rule. Finally, we also introduce a **min** and **max** function, which return, from a set of path, the best and worst path respectively within this set. When considering BGP routes, these paths are equal to the routes having the smallest and the greatest vector of attributes when ordered lexicographically.

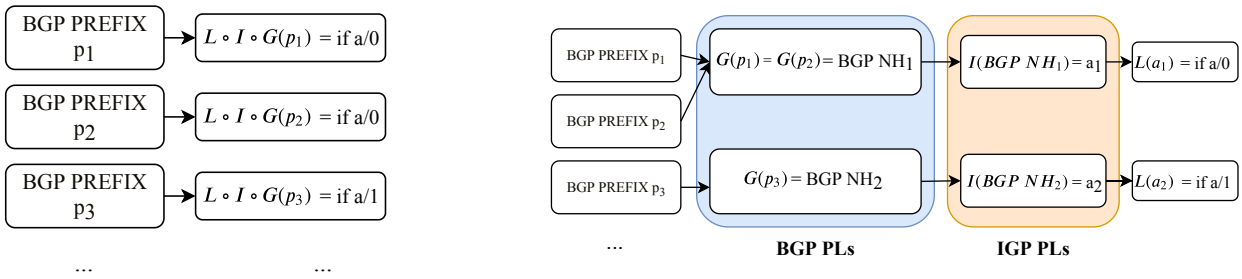
Using these notations, we can now detail one of the most simple implementation of the $G$ function. This implementation is described by the algorithm 3.1. We will not detail the inner working of the processing of a withdraw message, as it is not necessary to our study. We will however study in detail how our solution, OPTIC, handles these withdraw messages.

Listing 3.1: BGP best route update pseudo-code

```
Entry:
    r^new a newly received BGP route towards a destination r_d^new,
      composed of the route's attribute r_a^new=(r_a[LP], r_a[AS], r_a[I], r_a[TB]),
      the route's associated gateway r_g^new, and the route's
      destination r_d^new

    𝕊 a meta-set containing, for all destinations,
      the (optimal ECMP}routes 𝕊_o(d) for the destination d ,
      the associated (gateways 𝕊_g(d) ,
      and the set of routes saved 𝕊_w(d) in case of (withdrawal,
      of the chosen route, we construct these sets such that
          𝕊_w ∩ 𝕊_o = ∅
Output:
    Updated 𝕊


---

BGP_update_best_route(r^new, 𝕊):
  r^best = min(𝕊_o)
  IF r^best = ε OR r_a^new ≺ r_a^best: #compare new route to current best route
    𝕊_w = 𝕊_w ∪ r^best   #if new route is better, store current route away
    𝕊_o = r^new      # replace best routes
    𝕊_g = r_g^new   # replace best gateways
  ELSE IF r_a =_TB r_a^best: #ECMP
    𝕊_o = 𝕊_o ∪ r^new      # add to best routes
    𝕊_g = 𝕊_g ∪ r_g^new   # add to best gateways
  ELSE:   #r^best ≺ r^new
    𝕊_w = 𝕊_w ∪ r^new  # if new route is not better, store it away
```

Standard BGP path selection relies on keeping the best path $r^{best}$ towards any destination $d$. The most basic version of BGP keeps only the single best route. The version that is presented here is more advanced, as it keeps a set of ECMP routes, i.e., routes that are tied all the way up to the tie-break attribute (not included). This allows BGP to perform load-balancing across these various paths and potentially to find a fail-over path quickly if needed. However, the requirements necessary for two routes to be considered equal is quite heavy. In addition, while several routes are memorized, only the best one is advertised to peers if BGP add-path is not used. Other routes are kept within $\mathbb{S}_w$, and can be retrieved if the best routes ares withdrawn. Different paths (represented by a vector of attributes $(r_a[LP], r_a[AS], r_a[I], r_a[TB])$) which can be used to reach the same destination can always be compared, as BGP creates a strict total order between routes thanks to an arbitrary `Tie-Break` attribute. Thus, in practice, any two routes can be compared via the operator $\prec$. In some cases, namely if the MED attribute is used, this total order may be broken. This is not, however, one of the aspect that we will study here. One of the main issues regarding BGP lies in the way the routes are stored in standard FIB.

Routers usually rely on a flattened Forwarding Information Base , meaning that the FIB stores, for each destination, the result of the composition of functions explained above. Consequently, if a failure occurs and the current best BGP routes become unavailable or change, the BGP decision process must be re-run for each destination, in order to find the new result of the $PORT(d)$ function. More precisely, the decision process goes through the new informations in the Loc-RIB, in order to select the best path




(a) Flat FIB architecture

(b) Hierarchical FIB architecture

Figure 13: Flat Forwarding Information Base compared to Hierarchical Forwarding Information Base

for every prefix. The new path are then uploaded to the FIB and the Adj-RIB-Outs. This update is time-consuming and scales with the number of entries (prefixes) in the BGP RIB. Note that this may be triggered by a BGP update or an intra-domain failure. Indeed, while the IGP convergence is very quick, the preferred BGP next-hop may change due to hot-potato routing. For example, if an AS has three eBGP sessions with different ASes, an IGP event could change the best gateway for up to 300000 prefixes. During this convergence time, users may experience packet loss for up to 100 seconds. This convergence time, in addition to the time needed for the control-plane messages of BGP to propagate, creates a delay which is not viable for modern applications with strict quality of service requirements. While the solutions explored in Section 2.4 try to find ways to bypass the propagation time of BGP control plane messages, BGP Prefix Independent Convergence **(PIC)** aims at bypassing the convergence time of BGP.

## 2. PIC

BGP PIC, proposed by Filsfils et al. [6], aims at reducing the convergence time of BGP by modifying the way the forwarding information is stored within routers. While routers initially used a basic list storing a one-to-one relation between a destination and the corresponding outgoing interface, this simple architecture quickly becomes inefficient due to the large number of prefixes this architecture learned through BGP. While this table can be reduced quite efficiently, particularly when the routing policies are isotone [32], another possible solution is to use hierarchical FIBs, which allow to efficiently update routing information while maintaining all of the information learned through BGP. PIC proposes to use a hierarchical FIB **(HFIB)** to greatly speed up the recovery time and update time. This idea is further enhanced by extending the standard BGP ECMP set to two distinct ECMP sets to cover a broader array of failures. The basic idea behind the use of an HFIB is illustrated by Figure 13.

### 2.1 Hierarchical FIB

A hierarchical FIB (HFIB) allows the convergence time to be independent from the number of destinations stored within it, as it is not necessary to go through each and every destination. The usual flattened design is thrown away, to prefer this more complex architecture, which PIC in turn couples with an alternate route mechanism. In a flat FIB, a change to a BGP NH may in turn result in the need to update all the associated prefixes one by one. Note that, since the outgoing interface is the only information kept, even if the BGP NH stays the same but the outgoing interface to reach it changes, every prefixes still need to be updated and the FIB still needs to be walked through entirely.

Conversely to its flattened counterpart, an HFIB keeps the parent-child relationships between a BGP NH and the outgoing interface (i.e., the results of each function of the composition evoked in 1) within the FIB. While this architecture requires more memory accesses, as the chain $L \circ IH \circ G \circ P(d)$ needs to be gone through to find the outgoing interface, as well as more storage, the update time of forwarding information is drastically reduced, as a single modification of the result of $I$ or $G$ is reflected on the rest of the chain. More precisely, an IGP event that does not affect the BGP NH (i.e., a change of IGP NH or of the outgoing interface), does not require to walk the entirety of the BGP prefixes as in a flat FIB, but only the IGP prefixes, which are usually far less numerous. For example, let us imagine that upon a core failure, the IGP NH $a_1$ can no longer be reached through interface a/0. Once the IGP converges, we learn that $a_1$ should now be reached through interface a/2. Following the architecture described by Figure 13b, a single modification of the $L(a_1)$ entry is enough to restore connectivity to this node. If a standard flat FIB was used, each entry would be updated one by one.

## 2.2 Protection against core and edge failures

PIC tries to efficiently protect traffic from both *core* and *edge* failures. A core failure is defined as the failure of a link or node within the AS, excluding edges node. Conversely, an edge failure is defined as the failure of either a BGP peering link or of an edge node. In practice, PIC extends the basic architecture of an HFIB by creates *Path Lists* to store ECMP sets of various routes instead of a single route, to allows for quicker recovery upon failure. PIC first creates BGP Path lists, which are composed of two distinct ECMP sets of BGP NH for a destination, a primary set and a backup set. Each BGP NH will, in turn, point to a *IGP Path List*. As the name suggests, IGP **PLs** work is a similar fashion as BGP PLs, and contain, for each parent BGP NH, an ECMP set of IGP paths towards the gateway. Finally, each IGP NH within the set points towards an outgoing interface.

As stated above, to benefit as much as possible from the quick update time made possible by this architecture, PIC stores several routes for each destination. The primary set is composed of the best routes towards the destination, along with its ECMP alternates if possible. Similarly, the backup set is composed of the second to best routes and their ECMP alternates. The ECMP alternates exist only if a BGP multi path policy is used, which is usually not the case. Consequently, in practice, both sets will often contain only one route each. In other words, for any routes $r^1$,$r^2 \in \mathbb{S}_o$ and any routes $r^3$,$r^4 \in \mathbb{S}'_o$, then:

- $r^1_a \prec r^3_a$;
- $r^1_a =_{TB} r^2_a \equiv (r^1_a[LP], r^1_a[AS], r^1_a[I]) = (r^2_a[LP], r^2_a[AS], r^2_a[I])$: $r^1$ and $r^2$ are equal up to their tie break attribute;

  $r^3_a =_{TB} r^4_a$: $r^3$ and $r^4$ are equal up to their tie break attribute.

The construction of the primary and secondary sets of BGP NH can be described by Listing 3.2. At its most simple state, i.e., when no BGP multi-path policies other than ECMP BGP are used, PIC can be seen as a wrapper that create, for a destination $d$, two ECMP sets of route by re-running the standard BGP best path selection procedure a second time. More precisely, PIC maintains two meta-sets, $\mathbb{S}$ and $\mathbb{S}'$. $\mathbb{S}$ contains the result of the standard BGP algorithm as described by Listing 3.1. $\mathbb{S}'$ contains the result of the same BGP algorithm when run on the routes that are not contained within the best routes $\mathbb{S}_o$.

These two sets theoretically allow PIC to protect traffic from both core and edge failures. Indeed, upon a core failure, $L \circ IH(G)$ is recomputed to find the new best path and outgoing interface towards $G$. Once the modification is done and inserted in the corresponding IGP PL, the update will immediately be reflected onto each parent BGP PL. Consequently, a single operation is enough to restore the connectivity to the affected prefixes. Upon an edge failure, PIC relies on the two sets $\mathbb{S}_o$ and $\mathbb{S}'_o$ to restore connectivity quickly. Indeed, these two sets ensure that at least two paths are always known towards any given destination. An edge failure may result in the deletion of an IGP PL, as either the corresponding gateway itself is now unreachable, or the interface connected to the eBGP peer, which may be announced within the IGP, is now down. This deletion triggers an update which in turn modifies the depending BGP PL, and switch to a still reachable route within one of the two ECMP sets. The new route is either another route from $\mathbb{S}_o$ if a route is still reachable, or a route from $\mathbb{S}'_o$ otherwise. As this update is done for each BGP PL that needs to be changed after the failure, the update time scales with the number of BGP PL, which is usually notably less than the number of prefixes. In both cases, if the failure is within the AS, PIC is be able to use the fast response time of IGPs to update the RIB of the router quickly and in an efficient way.

Listing 3.2: PIC best routes update pseudo-code

```
Entry:
    r^new a newly received BGP route towards a destination d,
    characterized as a list of attributes (r_a[LP], r_a[AS], r_a[I], r_a[TB])

    S, S' sets containing, for all destinations d,
    an ECMP set S_o of the best received routes,
    an ECMP S'_o of the second to best received routes
    with their associated gateways S_g and S'_g
    and S_w, S'_w the set of all the unused routes.

Output:
    Updated S, S'
```

```
update_best_route(r^new, S, S'):
  old_set = S
  BGP_update_best_route(r^new, S) #update S with new best route
  IF old_set ≠ S: #S was changed
    IF |S| = 1: #S now contains only the new route, i.e, r^new ≺ old_set
      #As old_set will become the S', we need to remove it from the
      #withdrawn subset. S' becomes unused.
      S_w = (S_w \ old_set) ∪ S'
      S'_w = S'_w ∪ S'
      S'_o = old_set # the former best set becomes S'
    #else, the new route was simply added to the best set, i.e,
       r^new =_TB S
  ELSE: #S wasn't changed
    BGP_update_best_route(r^new, S') #update S'
    #The withdraw subset was updated, containing
    #now either the new route
    #or the old second best set. We simply need to update S
       accordingly.
    S_w = S_w ∪ S'_w
```

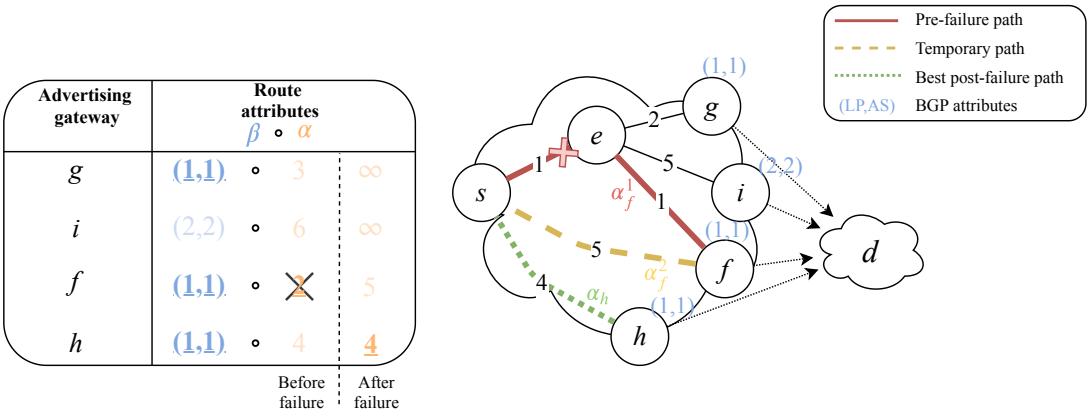



Figure 14: PIC may use a suboptimal path temporarily upon a core failure, as in intra-domain failure may change the BGP ranking of known routes.

## 2.3 Drawbacks

However, upon core failure, while PIC does indeed allow the connectivity to be restored quickly to prevent packet loss, it may sacrifice path optimality. Figure 14 illustrates such a case. Simply put, in Figure 14, $f$ and $g$ are two equally good gateways to reach $d$. Gateway $f$ is chosen due to its lesser IGP distance from $s$. Upon the failure of the link $s \rightarrow e$, $h$ becomes closer to $s$ than $f$ and should thus be used as the new gateway. However, PIC does not take this BGP-IGP dependency into account simply finds another path to $f$, consequently using a non-optimal path towards $d$.

More precisely, let $r^f$ and $r^h$ be two paths, advertised respectively by nodes $f$ and $h$ towards the destination prefix $d$, such that $r_a^f =_I r_a^h$ (i.e.,$(r_a^f[LP], r_a^f[AS]) = (r_a^f[LP], r_a^h[AS])$) and $r_a^f[I] < r_a^h[I]$. Consequently, $r_a^f \prec r_a^h$ and $G(d) = f$. After the failure between $s$ and $e$, the path towards f is recomputed, changing from $\alpha_f^1$ to $\alpha_f^2$, restoring the connectivity towards $d$. However, this new path has a different IGP distance. As shown in Figure 14, $\alpha_f^1$ and $\alpha_f^2$ have two different distances, and $\alpha_f^1 < \alpha_f^2$. More specifically, we can also now state that $\alpha_f^2 > \alpha_h$. We can thus update the equalities mentioned above: $r_a^f =_I r_a^h$, and $r_a^f[I] > r_a^h[I]$, resulting in $r_a^h \prec r_a^f$.

This case is ignored by PIC, which finds a new path towards the pre-failure gateway without considering that the failure affected the outcome of BGP decision process by altering the `IGP cost` $\alpha$ . In other words, PIC recomputes $L \circ IH(f)$, which is enough to restore connectivity, but $G(d)$ should be recomputed as well to ensure the optimality of the new path. Note that this change is temporary, as the BGP decision process will eventually be run again, allowing $G(d)$ to be recomputed and the new optimal path to be found. However, during this long convergence time [6], the impacted traffic will follow a suboptimal route, potentially leading to SLAs violations or congestions. In addition, in some particular topologies, PIC may fail to find a new path to begin with, either upon core or edge failures. Indeed, it is possible for a failure to render both of PIC's ECMP sets of route unreachable, depending on the network topology. In summary, depending on the type of failure and on the topology, PIC may either:

- Make the transiting traffic go through a non-optimal paths;
- Fail to find a backup path quickly;
- Disrupt flows by switching temporarily to a path before switching to the optimal path.

OPTIC (Optimal Protection Technique for Intra-domain Convergence) aims to solve these issues. ensures that, upon intra-domain failure, the new optimal path may always

be found quickly, allowing to protect the transiting traffic from any intra-domain failure in an optimal fashion.

# Chapter 4 Contribution: OPTIC

OPTIC tries to correct the flaws of PIC by modifying the $G$ function. OPTIC will neither store a single route per destination nor will it create two ECMP sets. Instead, OPTIC will create a single set of BGP NHs (or gateways) per destination, which we call a *1-optimal-protecting set*. We will start this section by formalizing our solution. We will introduce several ways to optimally protect transiting traffic from internal failures, which rely on more or less complex computations of 1-optimal-protecting sets.

## 1. Theory

**Definition 1.** *1-optimal-protecting (**1OP**) sets*
*A set $\mathcal{G}$ is said to be 1-optimal-protecting (1OP) for an external destination $d$ if:*

- *(i) $\mathcal{G}$ contains the best gateway $b$ towards $d$, with $\alpha_b$ as its best intra-domain route.*
- *(ii) For any failure of a component $c$ (node or link, $b$ included) such that $c \in \alpha_b$, the gateway of the new best path towards $d$ after the failure of $c$ is in $\mathcal{G}$.*

We state that a 1-optimal-protecting set ensure both the *protection* of a destination and the *optimality* of the new route chosen after an intra-domain failure. More precisely, these sets are *protecting* as they contain another available alternative route to their associated destination whatever the intra-domain failure. They are *optimal* as they contain the new best route towards their associated external destination after an intra-domain failure. Note that it is equivalent to consider a 1OP set of routes $\mathcal{O}$ or the 1OP set $\mathcal{G}$ of the associated gateways. Said otherwise, after any intra-domain failure, the new best post-convergence BGP NH towards $d$ can be found in $\mathcal{G}$. It is important to note that these sets are computed using the router's knowledge of the network. If this knowledge is too limited or incomplete, OPTIC cannot ensure that 1OP sets can be constructed. We thus assume that the BGP routes that are known are *sufficient* to create 1OP sets. Thus, in practice, OPTIC simply has to go through the 1-optimal-protecting set $\mathcal{G}_d$ of each destination $d$ to find the new best gateway, i.e., $\mathbf{min}(\mathcal{G}_d)$, to ensure an optimal recovery of the transiting traffic towards $d$ instead of having to wait for the BGP control-plane to converge. OPTIC can work with other existing FRR mechanisms to find the new best route upon internal failure, as detail in Appendix B.

As an example, in Figure 14, while PIC creates two distinct ECMP sets $\mathbb{S}$ and $\mathbb{S}'$, such that $\mathbb{S}_o = \{f\}$ and $\mathbb{S}'_o = \{g\}$, OPTIC creates a single set $\mathcal{G} = \{f, g, h\}$, where $f$ is the best gateway, $g$ would be the new best gateway if $e \to f$ fails, and $h$ would be the new best gateway if the node $e$ or the link $s \to e$ fails. The set constructed by OPTIC contains the new best gateway towards $d$ whatever the failure that occurs, as long as the failure only impact one component.

A 1OP set $\mathcal{G}$ has to contain at least two elements to protect the transiting traffic. Indeed, even if $b$ remains the optimal gateway whatever the failure, the failure of $b$ itself also has to be taken into account. As $b$ can't protect itself, at least one other gateway has to be added to $\mathcal{G}$ to ensure the protection of the traffic. It is thus impossible for a set to be 1OP if it contains less that two routes or gateways.

A set composed of all possible gateways (for example, all the border routers of the network) would still, by definition, be a 1-optimal-protecting set. However, using such sets would be fairly ineffective as a lot of useless information would have to be gone through when trying to find the new best gateway after a failure. We thus introduce the notion of minimal 1-optimal-protecting sets.

**Definition 2.** *Minimal 1-optimal-protecting sets*
*A 1-optimal-protecting set $\mathcal{G}$ is minimal if, after the removal of a single element of $\mathcal{G}$, $\mathcal{G}$ is not optimal anymore.*

Simply put, minimal 1-optimal-protecting sets protect a destination in an optimal fashion with the minimum number of gateways. [1] *Minimal* 1-optimal-protecting sets are theoretically the ideal way to use OPTIC, as they contain the bare minimum of gateways to ensure optimality and protection. As will be explain later on, OPTIC only maintains a set of unique 1OP set to greatly reduce the update complexity. We will show further in this thesis that using decreasing the size of 1OP sets decreases the number of unique 1OP sets, making the use of minimal 1OP set the theoretical ideal choice to improve OPTIC performance.

Indeed, as the new best gateway for each destination after a intra-domain failure can be found by going through each 1OP sets, the complexity of this update scales with the number of sets as well as their size. Since the number of sets mainly depends on the routes themselves and therefore can not be directly reduced, minimizing the size of 1OP sets is the most interesting way to reduce the complexity of OPTIC. However, using minimal 1OP sets can introduce an important overhead in the computation of the sets. We aim at finding a compromise between the minimality of 1OP sets and the complexity of their computation. As a result, we introduce the notion of **relaxed 1OP sets** . A relaxed 1OP set can be described as any 1OP set that is not minimal. The relaxed 1OP sets that we will be focusing on heavily rely on *IGP rounded sets* which we introduce in Definition 3. As a reminder, we use the representation of BGP routes introduced in our model, meaning that we consider a route $r$ as a triplet containing a destination $r_d$, a vector of attribute $r_a$ and a gateway $r_g$.

**Definition 3.** *IGP rounded sets*
*A set $\mathcal{G}$ is IGP rounded if it contains the gateways of every routes towards the same destination that share the same BGP attributes up until their IGP distances, and no other gateway.*
*That is, with $\mathcal{G} = \{r_g^1, r_g^2, ...r_g^n\}$ we have $r_a^1 =_I r_a^2 =_I ...r_a^n$ and for all routes $r$ such that $r \notin \mathcal{G}$, $r \neq_I \mathcal{G}$.*

An IGP rounded set is composed of all the routes (or associated gateways) that were discriminated only by their IGP distance, i.e, with our simplification, routes with equal local-pref and as-path attributes. In other words, when considering attribute as the composition of inter-domain related attribute and IGP related attributes $\beta \circ \alpha$, routes within the same IGP rounded set share the same $\beta$ attribute but may have a different $\alpha$ attribute. IGP rounded sets are a core concept of OPTIC, as they maintain their IGP rounded property after an intra-domain failure. This will allow us to create 1OP sets without computing the new shortest intra-domain path for every failure. Indeed, as an intra-domain may only change the IGP-cost attribute $\alpha$ of BGP routes and not $\beta$, the best $\beta$ of the BGP routes will remain the same after the failure, meaning that the new

[1]. The gateways composing a minimal 1OP set may, for example, be extracted by running one shortest path computation for each possible failure on the intra-domain route towards the best gateway

best route has to have the same $\beta$ as the former best route, and thus is within the same IGP rounded set.

Said otherwise, for two routes $r'$ and $r''$ such that $r'_a =_I r''_a$, the equality $r'_a =_I r''_a$ holds true after an intra-domain failure. Consequently, even though the BGP ranking of the routes that are within the same IGP-rounded may change due to an intra-domain failure and the modification of their IGP distance $\alpha$, they will still remain within the same IGP-rounded set. This equality hold true as each of our operators $\prec_I, =_I, \succ_I$ are unaffected by an IGP failure, as described more formally by Lemma 1.

---

**Lemma 1.** *The order given by the operators $\prec_I$, $=_I$ and $\succ_I$ is not affected by an intra-domain failure.*

---

As an intra-domain failure may only affect the `IGP cost` attributes $\alpha$ of a BGP route, which is not taken into account by $\prec_I$, $=_I$ and $\succ_I$, the proof of this Lemma is trivial. Lemma 1 is one of the keystones of our reasoning, as it allows us to state that the way we rank routes and the inequalities we create using our operators will not be modified by a failure. More precisely, when combined with the property of IGP rounded sets, we can state that while an intra-domain failure may change the BGP ranking of routes that are within the same IGP-rounded set, the ranking of the IGP-rounded sets themselves will remain unaffected by the failure. Simply put, routes within the $x^{th}$ IGP-rounded set will still be better than the routes within the $x + 1^{th}$ IGP-rounded set after a failure. Indeed, two distinct IGP rounded sets $\mathcal{G}'$ and $\mathcal{G}''$ satisfy either the inequality $\mathcal{G}' \prec_I \mathcal{G}''$ or $\mathcal{G}' \succ_I \mathcal{G}''$ by definition, and only the modification of the local-pref or the as-path attribute (which can not be changed by an intra-domain failure) may change these inequalities. As a result, since we only consider the local-pref, as-path, IGP distance and tie-break attributes, we know that the best route after a potential IGP distances modification due to an intra-domain failure is within the best IGP-rounded set.

It is however possible for routes within the best IGP rounded set to all be unusable after a failure, particularly if the best IGP rounded set only contains only a single route. In this case, we know that the second best route has to be in the second best IGP-rounded set, and so on. This idea is the core concept of OPTIC. Nevertheless, the number of IGP rounded sets that needs to be maintain depends on the network. As OPTIC introduces a lot of new concepts, we will slightly simplify our proofs, theorems, and definition by introducing the following hypothesis to better understand the inner working of OPTIC. This hypothesis will be lifted later on.

**Hypothesis 1.** *Biconnected graph: The underlying graph of the network is biconnected.*

In a biconnected graph, the graph stays connected, i.e., there exists a path between every pair of vertices, after the removal of any one vertex or edge. Networks being usually designed to be fairly resilient, it is not unlikely for this condition to be verified. However the point of this hypothesis is mainly pedagogical. We can and will adapt our algorithms and structures to lift this hypothesis later on, once the main ideas lying behind OPTIC are explained. One of the consequences of Hypothesis 1 is that a simple intra-domain failure cannot make more than one gateway unreachable. Indeed, if the failure does not affect the gateway itself, then there still exists a path towards the gateway. This hypothesis makes the protection property of 1OP set easy to achieve, as having two

gateways within a 1OP set is now enough to protect the destination. This will allow us to focus on and prove the method allowing to ensure the optimality of the set (i.e, that the set contains the new best route upon failure), which was introduced when defining IGP-rounded sets.

**Remark 1.** *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two sets such that for every gateways $r_g^1 \in \mathcal{G}_1$ and $r_g^2 \in \mathcal{G}_2$, we have $r_a^1 \prec_I r_a^2$. For readability purposes, we write that $\mathcal{G}_1 \prec_I \mathcal{G}_2$.*

As mentioned, IGP rounded sets allow us to introduce relaxed variants of 1OP sets, whose constructions are far less costly than minimal 1OP sets. In particular, OPTIC, as presented in Algorithms 4.1 and 4.2, uses the variant we call 2-rounded sets, which rely on Hypothesis 1.

**Definition 4.** *2-rounded **(2R)** sets*
*An efficient 2-rounded (2R) set $\mathcal{O}$ is composed of the two best routes $r^{best}$ and $r^{scnd}$, in addition to all the routes $r$ such that $r =_I r^{scnd}$, i.e., $\mathcal{O} = r^{best} \cup r^{scnd} \cup \{r | r_a =_I r_a^{scnd}\}$*

*Efficient 2R sets can be equivalently described as follows:*
*An efficient 2-rounded (2R) set $\mathcal{O}$ is composed of the union of the best IGP rounded sets of route, in order, up until $|\mathcal{O}| >= 2$. Formally, $\mathcal{O}$ can be decomposed as $\bigcup_{i=1}^{n} \mathcal{O}_i$, where $\mathcal{O}_n$ is an IGP rounded set, $\mathcal{O}_n \prec_I \mathcal{O}_{n+1}$, and $|\bigcup_{i=1}^{n-1} \mathcal{O}_i| < 2$*

*Any set containing more routes than evoked above is 1OP, but is said to be inefficient.*

Less formally, Definition 4 means that an efficient 2R set can be one of two possibilities. It is either composed of the best IGP rounded set if it contains at least two elements, or the two best IGP rounded sets. Thanks to the biconnected property, we never need to rely on more than two IGP rounded sets.

The best way to understand 2-rounded (2R sets) is to see the addition of the first two best routes $\{r^{min}, r^{scnd}\}$ to the set as a means to ensure the protection of a destination $d$ (two gateways being enough thanks to Hypothesis 1).

However, as an intra-domain failure may change the BGP ranking of the routes that are tied up until their IGP-cost attribute, we need to ensure that these routes are within the set as well, so that the set is optimal. Consequently, we IGP-round the set by adding each of these routes, i.e, all the routes $r$ that satisfy $r_a =_I r^{scnd}$. 2-rounded sets are a fairly good approach at creating 1OP sets, as they rely on light computations but still prune some unnecessary routes when they are efficient. Indeed, while a 1OP set is not necessarily 2-rounded, a 2-rounded set is necessarily 1OP.

To ensure that our version of OPTIC that uses efficient 2-rounded sets creates and maintains 1OP sets, we will start by showing that 2-rounded sets are indeed part of the family of relaxed 1OP sets by proving Theorem 1. Afterward, we will show that OPTIC, as described by the algorithms found later in this report, maintains 2R sets.

**Theorem 1.** *Let $\mathcal{G}$ be a 2-rounded set of gateways towards a destination $d$. If Hypothesis 1 is verified, then $\mathcal{G}$ is 1-optimal-protecting.*

*Proof.* **Protection**

Let $g$ be the best gateway towards a given destination $d$. A single gateway is enough $d$ against the failure of any component that is not $g$, thanks to Hypothesis 1. To protect $d$ against the failure of $g$ itself, we need to add a second gateway $f$. As we now that $f$ will still be reachable if $g$ fails, given Hypothesis 1, having two gateways within our set is enough to ensure protection against any intra-domain failure. By definition, a 2R set contains 2 elements and ensures thus the protection of the destination.

**Optimality**

We will prove the optimality property by contradiction. Let $\mathcal{G}$ be a 2R set for a destination $d$. Let $r^{contr}$ be the new best route towards $d$ after an intra-domain failure, such that $r_g^{contr} \notin \mathcal{G}$. If $\mathcal{G}$ is a 2-rounded set, then $\mathcal{G}$ can be written as $r^{best} \cup r^{scnd} \cup \{r | r_a =_I r_a^{scnd}\}$ , where $r^{best}$ and $r^{scnd}$ were the two best routes during the construction of the set, according to Definition 4.

Thus, if $r_g^{contr} \notin \mathcal{G}$ and $r^{contr}$ was already known during the construction of the set, then $r_a^{contr} \succ_I r_a^{scnd}$ by definition. Since, thanks to Hypothesis 1, at least one of the two gateways $r_g^{best}$ or $r_g^{scnd}$ will still be reachable after failure, and that we showed that $r_a^{best} \preccurlyeq_I r_a^{scnd} \prec_I r_a^{contr}$, then there will still exist a route $r^x \in \mathcal{G}$ such that $r_a^x \prec_I r_a^{contr}$ after failure, as this order will not be altered, according to Lemma 1. Thus, $r_g^{contr}$ cannot be the new best gateway.

As the new best gateway upon failure is within $\mathcal{G}$, the set is optimal. ■

## 2. Algorithms

As explained above, this version of OPTIC uses efficient 2R sets to maintain and create 1OP sets for each destination, when possible. Algorithm 4.1 explains how OPTIC treats an update message from BGP.

As showcased by Algorithm 4.1, OPTIC maintains a meta-set $\mathbb{S}$ containing, for each destination $r_d$, three sets $\mathbb{S}_o(r_d)$, $\mathbb{S}_g(r_d)$, $\mathbb{S}_w(r_d)$. $\mathbb{S}_g(r_d)$ is a 1OP set of gateways towards $r_d$, $\mathbb{S}_o(r_d)$ is the set of associated route from which the gateway composing $\mathbb{S}_g(r_d)$ were extracted, and $\mathbb{S}_w(r_d)$ contains the routes that are not needed to form a 1OP set. This set will be useful if a gateway is withdrawn, and OPTIC needs to fetch one of the unused route to restore the 1OP property.

The main idea behind OPTIC is that, upon the reception of a new route, OPTIC will compare the route to the ones stored in the 1OP sets. If the route is not needed to ensure the 1OP property, it is stored away in $\mathbb{S}_w(r_d)$. Otherwise, it is added to $\mathbb{S}_o(r_d)$.

To simplify our algorithms for readability purposes, we consider that OPTIC maintains one 1OP set per destination. In reality, 1OP sets can be shared by prefixes. In practice, one can imagine our set of gateways $\mathbb{S}_g$ as a pointer, or a hash index, referencing an index in a structure containing all the existing 1OP sets of gateways. Two destinations sharing the same 1OP set of gateways point towards the same 1OP set. Allowing destinations that have the same 1OP set to share the same 1OP set structure reduces OPTIC complexity upon update, as the number of set is reduced and the modification of a set affects all the dependent destinations. This is further explained in Appendix C.

Listing 4.1: OPTIC route update pseudo-code

```
Entry:
    r^new a newly received BGP route towards a destination r_d^new,
      composed of the route's attribute r_a^new=(r_a[LP], r_a[AS], r_a[I], r_a[TB]),
      the route's associated gateway r_g^new, and the route's
      destination r_d^new

    𝕊 a meta-set containing, for each destination d,
      the 1-optimal-protecting set of routes for the destination 𝕊_o,
      the associated 1-optimal protecting set of gateways 𝕊_g,
      and the set of routes saved 𝕊_w in case of withdrawal
      of the chosen routes, such that 𝕊_w ∩ 𝕊_o = ∅

Output:
    Updated 𝕊
```

```
update_best_route(r^new, 𝕊):
  dest = r_d^new
  𝒢, 𝒪, 𝒲 = 𝕊_g(dest), 𝕊_o(dest), 𝕊_w(dest)
  r^best = min(𝒪); r^worst = max(𝒪)
  IF |𝒢| ≥ 2 AND r_a^new ≻_I r_a^worst: #non-optimal route 1+1
    𝒲 = 𝒲 ∪ r^new #store away
    RETURN
  ELSE IF r_a^new ≺_I r_a^best: #absolute best route
    IF r_a^worst =_I r_a^best: #set was IGP rounded
      𝒪 = 𝒪 ∪ r^new; 𝒢 = r_g^new ∪ 𝒢
    ELSE: # set was not IGP rounded
      𝒪' = 𝒪 ∪ r^new; 𝒪 = {r^new, r^best}; 𝒲 = 𝒲 ∪ 𝒪' \ 𝒪; 𝒢 = {r_g^new, r_g^best}
  ELSE IF r_a^worst =_I r_a^best: #=_I r^new: equivalent route within the IGP rounded set N ≥ 2
    𝒪 = 𝒪 ∪ r^new; 𝒢 = 𝒢 ∪ r_g^new
  ELSE IF r_a^new ≺_I r_a^worst: # unique second best
    𝒪' = 𝒪 ∪ r^new; 𝒪 = {r^best, r^new}; 𝒲 = 𝒲 ∪ 𝒪' \ 𝒪; 𝒢 = {r_g^best, r_g^new}
  ELSE IF r_a^new =_I r_a^worst: # add to second best
    𝒪 = 𝒪 ∪ r^new; 𝒢 = 𝒢 ∪ r_g^new
  𝒢, 𝒪, 𝒲 = 𝕊_g(dest), 𝕊_o(dest), 𝕊_w(dest)
```

In addition to adding necessary routes, OPTIC may *reconfigure* the 2R sets if some routes become useless after the reception of an update. Indeed, we want our 2R sets to stay efficient, which can result in the need to store some routes away. To show that OPTIC does not remove any necessary route nor does it add or keep any unnecessary route, we will prove the following property:

**Property 1.** *If Hypothesis 1 is verified and learned BGP routes are sufficient, then OPTIC creates and maintains efficient 2R sets for each destination d upon the reception of a BGP* `UPDATE` *message.*

*Proof.* Let us prove that OPTIC, as described by Algorithm 4.1 allows to create efficien 2R sets for a destination $d$, if enough routes towards it are known. Note that $\mathbb{S}_g(r_d)$ and $\mathbb{S}_o(r_d)$ follow the same evolution and can thus be seen as equivalent. To simplify the notations, let $\mathcal{G}$ be equal to $\mathbb{S}_g(r_d)$ and $\mathcal{O}$ be equal to $\mathbb{S}_o(r_d)$. We will use the comments in Algorithm 4.1 (`equivalent`, `non-optimal`...) as labels to reference the different cases we need to consider.

To be considered, a 2R set needs to **contain at least two gateways** and need to be able to be decomposed as $r^{best} \cup r^{scnd} \cup \{r | r_a =_I r^{scnd}\}$.

As shown by Algorithm 4.1, the `non-optimal` case can not be reached if $|\mathcal{G}| < 2$, meaning that OPTIC is bound to accept any route when $|\mathcal{G}| < 2$. Since all other cases add a route (`absolute best and rounded`,`equivalent`, `non-optimal`, `add to second bests`) or bring $\mathcal{G}$ back to two elements (`absolute best and not rounded`, `replace second best`), once $|\mathcal{G}| \geq 2$, the cardinal of the 1OP set can never go back to one, as we either store a new route away (keeping the cardinal of our set constant), add a route to our set, or reduce $\mathcal{G}$ to 2 elements

Consequently, as OPTIC accepts any route when $|\mathcal{G}| < 2$ and always keeps at least two gateways in the set afterward. **Therefore, the sets constructed by OPTIC always contain at least 2 gateways**.

We will now show that the sets constructed by OPTIC can always be decomposed as $r^{best} \cup r^{scnd} \cup r | r_a =_I r^{scnd}$. At first, when only two gateways are known by the router, OPTIC will create a set containing the two routes, as demonstrated above. Thus, our sets will be composed of two routes $r'$ and $r''$, such that $r'_a \preccurlyeq_I r''_a$. As $\{r', r''\}$ represents all the possible routes, there does not exit a route $r$ such that $r_a =_I r'_a$ or $r_a =_I r''_a$. Consequently, our set can be written as $r' \cup r'' \cup \{r | r_a =_I r''_a\}$, with $\{r | r_a =_I r''_a\} = \varnothing$ and $r'$, $r''$ the two best routes. Therefore, by definition, $\{r', r''\}$ is a 2R set. Since this set contains only two routes, it is efficient.

To show that OPTIC maintains this efficent 2R set, we will show that we only add necessary routes , and that we only remove routes that are useless to ensure the efficient 2R property. There are six different cases we need to address, referenced in Algorithm 4.1 as `absolute best and rounded`, `equivalent`, `add to second best`, `non-optimal`, `absolute best and not rounded`, `unique second best`. In each of these cases, we change our sets of routes and gateways. In the first three cases, we add only add a route to our set, while some routes may be removed in the other three *reconfiguring* cases. We will start by proving that the route we add during the three first cases are necessary, before showing that the routes we remove during the last three cases are unnecessary.

Let $\mathbb{S}(d)$ be the metaset containing, for a destination $d$, the 1OP set of gateways $\mathcal{G}$ and the associated set of route $\mathcal{O}$. Let $r^{new}$ be the route we are currently processing, i.e, the route we received in the update message. Let $r^{best}$ be equal to $\mathbf{min}(\mathcal{O})$ and $r^{worst}$ be equal to $\mathbf{max}(\mathcal{O})$.

The `absolute best and rounded`, `equivalent` and `add to second best` cases may be translated into the following inequalities respectively :

- $r_a^{new} \prec_I \mathcal{O}$, with $r_a^{best} =_I r_a^{worst}$. In this case, $r^{new}$ is the new best route. As a 2R set is, by definition, composed of at least the two best routes, $r^{new}$ indeed needs to be added to the set. The new set can now be written as $r^{new} \cup r^{best} \cup \{r | r_a =_I r^{best}\}$, with $r^{best} \cup \{r | r_a =_I r^{best}\} = \mathcal{O}$, respecting the definition of efficient 2R sets.

- $r_a^{new} =_I \mathcal{O}$. In this case, the 1OP set is composed of routes being tied at the IGP cost rule. If $r^{new}$ is not added, then, with $r^{scnd}$ the second best route of $\mathcal{O}$, there exist a route $r = r^{new}$ such that $r \notin \mathcal{O}$ and $r_a =_I r_a^{scnd}$, violating the definition of 2R sets. Consequently, $r^{new}$ needs to be added to the set. $\mathcal{O}$ can now be written as $r^{best} \cup r^{scnd} \cup \{r | r_a =_I r_a^{scnd}\}$, with $r^{new} \in \{r | r_a =_I r_a^{scnd}\}$, which respects the definition of efficient 2R sets.

- $r_a^{worst} =_I r_a^{new}$. In this case, as $\mathcal{O}$ is 2R, we know that, by definition, with $r^{scnd}$ the second best route in $\mathcal{O}$, $r^{worst} =_I r^{scnd}$. Consequently, if $r^{new}$ is not added, there

would exist a route $r = r^{new}$ such that $r \notin \mathcal{O}$ and $r_a =_I r^{scnd}$, which violates the definition of 2R sets. Consequently, $r^{new}$ needs to be added to $\mathcal{O}$. $\mathcal{O}$ can now be written as $r^{best} \cup r^{scnd} \cup \{r | r_a =_I r_a^{scnd}\}$, with $r^{worst}, r^{new} \in \{r | r_a =_I r_a^{scnd}\}$ which respects the definition of efficient 2R sets.

We are left with three cases: `non-optimal`, `unique second best` and `absolute best and not rounded`. Conversely to the cases above, these latter reconfigure our set, i.e, one or several routes may be removed. We thus need to ensure that the routes that are being removed are not necessary to maintain the 1OP property. As before, since we have shown that our set contains at least two routes, we have at least two routes in $\mathcal{O}$. These three cases can be translated into the three following inequalities respectively:

- $\mathcal{O} \prec_I r_a^{new}$. By construction, $\mathcal{O}$ is 2R and can thus be written as $r^{best} \cup r^{scnd} \cup \{r | r_a =_I r^{scnd}\}$, with $r^{best}$ and $r^{scnd}$ being the two current best routes. Consequently, if we add $r^{new}$, $\mathcal{O}$ would become $r^{best} \cup r^{scnd} \cup \{r | r_a =_I r^{scnd}\} \cup r^{new}$, as $r^{new} \notin \{r | r_a =_I r^{scnd}\}$ since $\mathcal{O} \prec_I r_a^{new}$. We can see that adding $r^{new}$ would make $\mathcal{O}$ inefficient as it would contain more routes than the definition of the 2R sets. We can thus store $r^{new}$ away.

- $r_a^{new} \prec_I r_a^{worst}$ and $r_a^{best} \neq_I r_a^{worst}$. Thus, $r_a^{best} \prec_I r_a^{new} \prec_I r_a^{worst}$. Here, $r^{new}$ is the new second best route, which means, by definition, that $r^{new}$ must be in $\mathcal{O}$. Since $r_a^{new} \prec_I r_a^{worst}$, $\mathcal{O}$ can now be written as $r^{best} \cup r^{new} \cup r^{worst} \cup \{r | r_a =_I r_a^{worst}\}$. We can see that $r^{best} \cup r^{new}$ is enough to form an efficient 2R set, as it is equal to $r^{best} \cup r^{new} \cup \{r | r_a =_I r_a^{new}\}$ with $\{r | r_a =_I r_a^{new}\} = \varnothing$ and with $r^{new}$ and $r^{best}$ being the two best routes. Consequently, keeping the two best routes is enough to maintain an efficient 2R set, as is done in our algorithm.

- $r_a^{best} \neq_I r_a^{worst}$ and $r_a^{new} \prec_I r_a^{best}$. Thus, $r_a^{new} \prec_I r_a^{best} \prec_I r_a^{worst}$. This case is identical to the case above, with $r_a^{new}$ and $r_a^{best}$ switched within the inequality. Applying the same reasoning, we know that we only need to keep the two best routes $r^{new}$ and $r^{best}$, as is done in our algorithm.

**Whatever the operation done by OPTIC, the sets constructed can always be decomposed as $r^{best} \cup r^{scnd} \cup \{r | r_a =_I r^{scnd}\}$.**

Consequently, we have shown that OPTIC starts by creating a 2R sets. Each modification done by OPTIC allows to maintains these efficient 2R sets. ■

Now that we showed that OPTIC can indeed maintain 2R sets upon the reception and processing of an UPDATE message, we need to show that the same can be said upon the reception of a WITHDRAW message. The behavior of OPTIC, in this case, is shown in Algorithm 4.2.

Listing 4.2: OPTIC route withdrawal pseudo-code

```
Entry:
    r^new a newly received BGP route towards a destination r_d^new,
      composed of the route's attribute r_a^new=(r_a[LP], r_a[AS], r_a[I], r_a[TB]),
      the route's associated gateway r_g^new, and the route's
      destination r_d^new

    𝕊 a meta-set containing, for all destinations,
      the 1-optimal-protecting set of routes for the destination 𝕊_o,
      the associated 1-optimal protecting set of gateways 𝕊_g,
      and the set of routes saved 𝕊_w in case of withdrawal
      of the chosen routes, such that 𝕊_w ∩ 𝕊_o = ∅

Output:
  Updated 𝕊
```

```
withdraw_route(r^new, 𝕊):
  dest = r_d^new
  𝒢, 𝒪, 𝒲 = 𝕊_g(dest), 𝕊_o(dest), 𝕊_w(dest)
  IF r ∉ 𝒪: #nothing to be done
    RETURN
  ELSE:
    𝒪 = 𝒪 \ r; 𝒢 = 𝒢 \ r_g
  IF 𝒪 ≥ 2: #Still 2R
    RETURN
  r^wmin = min(𝒲)
  FORALL r ∈ 𝒲: #Add necessary routes
    IF r_a =_I r_a^wmin:
      𝒪 = 𝒪 ∪ r; 𝒢 = 𝒢 ∪ r_g; 𝒲 = 𝒲 \ r
  𝒢, 𝒪, 𝒲 = 𝕊_g(dest), 𝕊_o(dest), 𝕊_w(dest)
```

The behavior of OPTIC upon the withdrawal of a route is pretty straightforward. If the route was not used in our 2R set, nothing has to be done. If our set still contains at least two elements after the deletion of the route, it is still a 2R set and nothing has to be done. Finally, otherwise, we restore the 2R property of our set by adding all unused route that satisfies $r_a =_I r_a^{wmin}$, i.e, the best route in $\mathbb{S}_w(r_d)$ with the associated IGP rounded set.

**Property 2.** *If Hypothesis 1 is verified and learned BGP routes are sufficient, then OPTIC creates or maintains 2R sets for each destination d upon the reception of a BGP WITHDRAW message.*

*Proof.* We have shown before that OPTIC starts by creating a 2R set and maintains it if possible. We can thus assume that upon the reception of the withdraw message, the set $\mathcal{O}$ was a 2R set. We will also assume that all unused routes were stored within $\mathcal{W}$, with $\mathcal{W} = \mathbb{S}_w(r_d)$.

Let us show that at the two critical points in our algorithm, `Still 2R` and `Add necessary routes`, the 2R property of $\mathcal{O}$ is still ensured.

– In the `Still 2R` case, we remove a route from a set that was 2R. Let us imagine that our set $\mathcal{O}$ lost its 2R property after the removal. Since, in the `Still 2R` case, $|\mathcal{O}| \geq 2$, we can decompose this set as $\{r^{best}, \mathcal{S}\}$ with $r_a^{best} \preccurlyeq_I \mathcal{S}$ and $|\mathcal{S}| \geq 1$. Thus, for $\mathcal{O}$ to not be 2R, then one of the following condition needs to be true:

(1): $r^{best}$ is not the best route, i.e, there exists a route $r$ such that $r_a \prec_I r_a^{best}$ and $r \notin \mathcal{O}$.

(2): $\mathcal{S}$ does not contain the second best route, i.e, there exists a route $r$ such that $r_a \preccurlyeq_I \mathcal{S}$ and $r \notin \mathcal{O}$.

(3): $\mathcal{S}$ is not IGP rounded, i.e, there exists a route $r$ such that $r_a =_I \mathcal{S}$ and $r \notin \mathcal{O}$.

However, a withdrawal message may only remove a route, and neither add nor change existing routes. Therefore, for the inequalities stated above to be true after the withdraw, these latter have to be true before the withdrawal as well. If these inequalities were true before the withdrawal, then $\mathcal{O}$ was not a 2R set, which contradicts our starting hypothesis. Consequently, if $\mathcal{O}$ is a 2R set before the processing of a withdraw message, $\mathcal{O}$ stays 2R after removing a route if $|\mathcal{O}| \geq 2$.

– In the `Add necessary route` case, after the removal of the route, $\mathcal{O}$ contains only one route $r^1$ and is thus not a 2R set. Let us show that adding each route $r$ that satisfies $r_a =_I r_a^{wmin}$, where $r^{wmin} = \mathbf{min}(\mathcal{W})$ is necessary to regain the 2R property.

We have shown above that it is not possible for a route $r$ to exist in $\mathcal{W}$ such that $r_a \preccurlyeq_I r_a^1$, i.e., no route in $\mathcal{W}$ are better or equal to the routes that were in $\mathcal{O}$, since $\mathcal{O}$ was 1OP. Thus, $r^{wmin}$ is the second best route and all the routes $r$ we will add satisfy the inequality $r_a^1 \prec_I r_a$. After adding these routes, we have $\mathcal{O} = r_a^1 \cup r^{wmin} \cup \{r | r_a =_I r_a^{wmin}\}$, with $r^1$ the best route, and $r^{wmin}$ the second best route, respecting so the definition of efficient 2R sets.

In both cases, the behavior of OPTIC maintains the property of the 2R set. ■

Having proven that both `UPDATE` and `WITHDRAW` messages are processed by OPTIC in a way that maintains the 2R property of the sets, we can now prove that OPTIC, as a whole, creates and maintains 2R sets, i.e., 1OP sets.

---

**Theorem 2.** *OPTIC creates and maintains 1OP sets if possible and if Hypothesis 1 is verified.*

---

*Proof.* We have shown thanks to Property 1 and 2 that OPTIC maintains 2R sets when a route is added and withdrawn. Thus, we only have to ensure that upon the processing of an update, OPTIC indeed saves the now unnecessary routes in $\mathcal{W}$, in case they might become necessary during the processing of a withdrawal.

We can, again, consider each case of the Algorithm 4.1 one by one. As some of the cases add new useful routes to our set, there is no need to update $\mathcal{W}$ in these cases. Only the `non-optimal` case, the `absolute best and not rounded` case, and the `unique second best` case should add routes to $\mathcal{W}$.

The unused `non-optimal` routes are explicitly added to $\mathcal{W}$. When some routes are removed, i.e., in the `absolute best and not rounded` and `unique second best` cases, we add to $\mathcal{W}$ the set $\mathcal{O}' \setminus \mathcal{O}$, where $\mathcal{O}'$ is equal to the routes that were in the

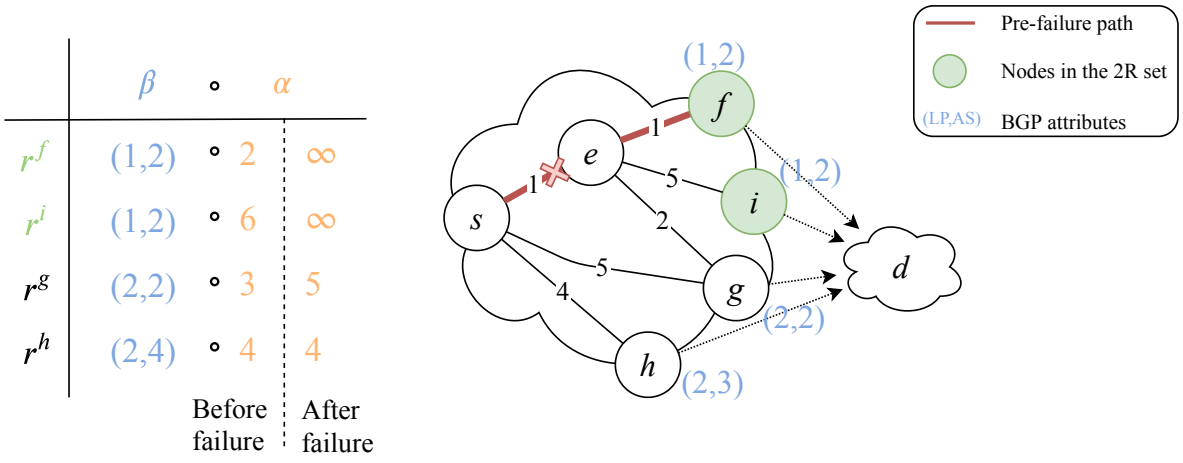



Figure 15: 2R sets and PIC problem with non-biconnected networks. A single failure renders the entirety of our 2R sets, or of the two ECMP set of PIC, unreachable.

old 2R set and the new route $r$, and $\mathcal{O}$ is equal to the now necessary routes in the 2R set. Therefore, $\mathcal{O}' \setminus \mathcal{O}$ contains the unnecessary routes in the set, that are then added to $\mathcal{W}$.

As OPTIC stores all the unused routes, maintains and creates 2R set upon addition and withdrawal of routes, OPTIC creates and maintains 1OP sets. ■

Consequently, the version of OPTIC that uses 2R sets, as described in this thesis, allow to protect any external destination from any simple intra-domain failure when the routes are sufficient by creating 1OP sets. In addition, we ensured the optimality of the new path chosen by OPTIC, i.e, OPTIC chooses the same path as the BGP decision process.

While this presentation of OPTIC was quite formal in order to prove our algorithms, we can also see OPTIC as an automata that make a set of routes transition between several states that represent 2R sets. This is further explained in Appendix D.

# 3. Improving OPTIC

## 3.1 Relaxing the biconnected hypothesis

While the biconnected hypothesis was useful to get a first grasp at the inner working of OPTIC, it is a fairly restrictive hypothesis. OPTIC can ensure the 1-optimal-protection of each destination even when the network does not respect this hypothesis if we slightly tune the definitions of the 2R sets we presented. First, it is useful to understand why removing the biconnected hypothesis increases the complexity of the problem we aim to solve. If a network is biconnected, we know that an internal failure may only make a single gateway unreachable, if the gateway itself fails. Any other failure may only impact the IGP distance $\alpha$ to each gateway. In particular, if our set contains $x$ gateways, at least $x - 1$ gateway(s) will still be reachable after the failure. This is not the case with non-biconnected networks.

Let us refer to Figure 15 to understand the problem, as it illustrates a non-biconnected network. This figure illustrates the same network as Figure 14, but with slightly different BGP attributes. Here, our 2R set would be composed of a single IGP rounded set $\{f, i\}$. Upon the failure of $s \to e$, we lose both of our gateways: we do not ensure the protection of $d$ upon any simple internal failure. Note that PIC also encounters a problem with this topology, as it relies on the IGP computing a new path to the best gateway $f$ after

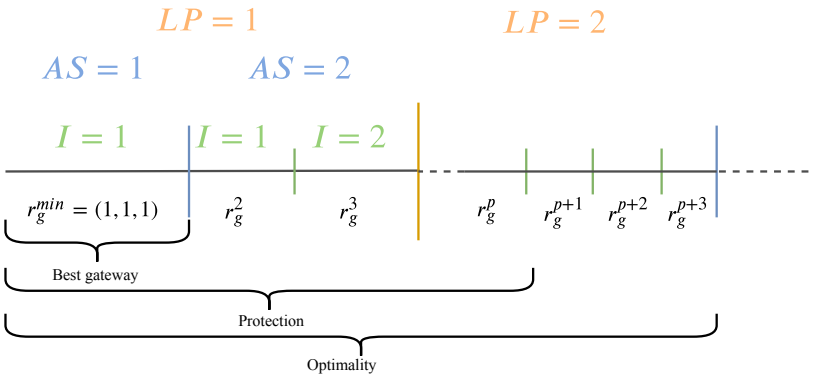



Figure 16: Construction of P-Rounded sets. We take as many gateways as necessary to ensure the protection of the destination, up until the gateway $r_g^p$. We then add every gateway $r_g$ such that $r_a =_I r_a^p$. 2R set are a particular type of PR set, where $p = 2$ is viable thanks to the biconnected hypothesis.

a core failure. However, here, there is no path towards this gateway. Even after $f$ is considered as unreachable, if we assume that PIC may rely on its ECMP backup set, $i$, this latter is also now unreachable. PIC has to wait for BGP to re-converge, and in the meantime, is unable to reach $d$.

Thankfully, we can extend 2R sets to cover non-biconnected networks and ensure the protection of $d$, by introducing P-rounded sets.

**Definition 5.** *P-rounded sets*
*A P-rounded set $\mathcal{O}$ is composed of the unions of the $P$ best IGP rounded sets of routes, in order, up until $|\mathcal{O}| \geq p$, where $p$ is the number of gateways that ensures that the destination $d$ is protected.*
*Formally, $\mathcal{O}$ can be decomposed as $\bigcup_{i=1}^{n} \mathcal{O}_i$, where $\mathcal{O}_i$ is an IGP rounded set, $\mathcal{O}_i \prec_I \mathcal{O}_{i+1}$, and $|\bigcup_{i=1}^{n-1} \mathcal{O}_i| < p$*

Less formally, this extension is fairly intuitive. The construction algorithm is illustrated in Figure 16. With biconnected networks, we knew that 2 gateways were enough to protect the destination. Thus, we protection as guaranteed by using two gateways, and the IGP rounding ensured optimality. Here, we need $p$ gateways to guarantee protection, and the IGP rounding to ensure optimality. In both cases, we can see the algorithm behind OPTIC as adding gateways until we protect the destination, then rounding the set at the IGP level.

Finding the value $p$ can be done through a linear test. For example, we can check for each component $c$ of the best internal path towards the best gateway if at least one of the gateways in our set is still reachable upon the failure of $c$ through a depth-first traversal. If not, we add the next best reachable gateway. If for any failure, at least one of the gateways in our set is still reachable, the set is 1OP.

Let us take an example following Figure 15. We start with a set containing the best gateway towards $d$, $\{f\}$ . Let $\alpha$ be the best path from $s$ to $f$. Let us check if a failure of a component of $\alpha$ make all the gateways in our set unreachable. If $e \to f$ fails, $f$ is not reachable. We add $i$, the next best reachable gateway, to our set in order to be protected from this failure: $\{f, i\}$ . If $e$ fails, neither $f$ nor $i$ is reachable: we add the next best reachable gateway $g$. If $s \to e$ fails, $g$ is still reachable through $s \to g$. We tested each component of $\alpha$. However, we can not end our algorithm here, as, after an internal failure, any gateway $k$ such that $k =_I g$ could be the best new gateway. Consequently,

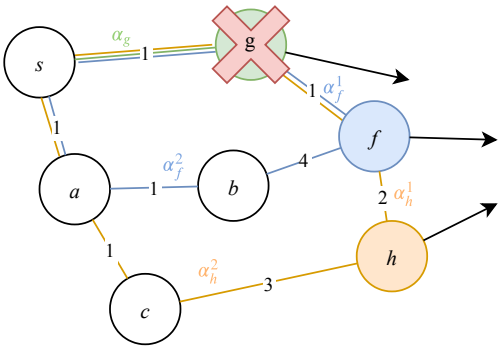



Figure 17: Dependent gateways. The failure of $g$ affects $\alpha_f$.

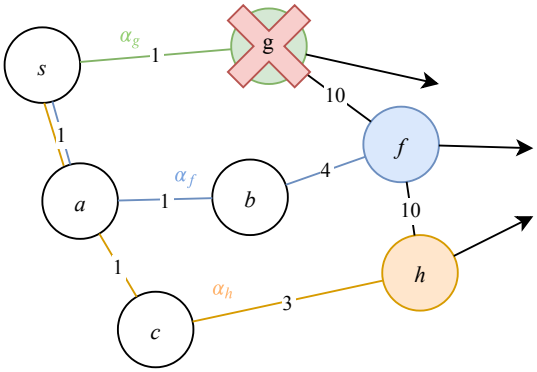



Figure 18: Independent gateways The failure of $g$ does not affect $\alpha_f$.

we add to our set all the gateways that respect this equality. In our figure, no gateway satisfies the inequality, leaving the final 1OP set as $\{f, i, g\}$.

While having a biconnected network is a sufficient condition for 2-rounded sets to be 1-optimal protecting, it is not a necessary condition. Indeed, we can refer back to Figure 14. While the network was not biconnected, a 2R set would have been 1OP. In practice, we can expect 2R sets to be 1-optimal-protecting most of the time. We could consequently take advantage of the simplicity of the computation of 2R sets, by starting to check if a 2R set is sufficient to ensure both the optimality and the protection of a destination $d$. To do so, we can start by building our 2R set, remove all the components of the best path $\alpha$ towards the best gateway $g$, and check if $g$ or the other gateways in our 2R set is still reachable. If it is, there is no need for further computation. If it is not, we can construct a 1OP set as evoked above. In most cases, we may profit from this greedy way of constructing 1OP sets. This may be especially useful as 2R sets can be improved depending on the topology of the network.

## 3.2 2R sets optimization

While we aim for 2R sets to be an easily computable variant of relaxed 1OP set, there is a slight optimization that can easily be done to reduce the size of the sets. This optimization lies in 2R sets in which $r_a^{best} \neq_I r_a^{scnd}$. when we needed the best gateway $r_g^{best}$ and the IGP rounded set $\mathcal{G}'$, with $r_a \prec_I \mathcal{G}'$ with $\mathcal{G}' =_I r^{scnd}$. There are some cases where we do not need this second IGP rounded set. As a reminder, we need the second IGP set $\mathcal{G}'$ to ensure optimality if the internal failure changes the IGP distance to the gateways in our sets. As mentioned, a failure will not change the order between two IGP rounded sets. Therefore, we need to use one of the gateway in $\mathcal{G}'$ if and only if $r_g^{best}$ is unreachable after the failure, as $r_a^{best} \prec_I \mathcal{G}'$.

However, the failure of $r_g^{best}$ will only affect the IGP distance of the gateways in $\mathcal{G}'$ if $r_g^{best}$ was part of the internal path towards the gateways of $\mathcal{G}'$. We refer to this case as *dependent gateways*. A gateway $f$ is dependent to a gateway $g$ if, $\alpha$ being the best path to reach $f$, $g \in \alpha$. Figure 17 illustrates dependent gateways. We can see that since $g \in \alpha_f^1$ and $g \in \alpha_h^1$, after the failure, the ranking of the route in term of IGP distance from $s$ goes from $f \prec g$ to $g \prec f$. We thus need IGP rounding to ensure that the new best gateway is in our set.

If the backup gateways are independent of the best gateway $r_g^{best}$, we know that if $r_g^{best}$ becomes unreachable, then the IGP distance of the other gateway remained the same. Consequently, the ranking of the gateways within $\mathcal{G}'$ is easily predictable. When

we need to fetch a gateway in $\mathcal{G}'$, then the best gateway within the set after the failure is the best gateway of the set before the failure. This case is illustrated in Figure 18. As $g \notin \alpha_f$ and $g \notin \alpha_h$, the ranking of the route in terms of IGP distance from $s$ stays $h \prec f$ before and after the failure. Consequently, we only need to memorize $h$, as $f$ will never be the best possible gateway: we do not need IGP rounding. Thus, if the network is biconnected and the second best gateway is independent of the best gateway, we only need to memorize the best and second best gateway to create a 1OP set.

In practice, this optimization can be implemented as a quick dependency test ensuring that the path to the second best gateway does not contain the best gateway. If it does not, the set is 1OP. If it does, the set needs to use IGP rounding.

This optimization allows to reduce the size of 1OP sets, which comes into play regarding the complexity of updating the best gateway upon and internal failure. As a reminder, destinations may share the same 1OP set, so that updating this set immediately affects all the associated prefixes. This aspect, while not necessary to understand OPTIC, is crucial in practice, as it allows to drastically reduce the complexity of the process needed to be done upon changes in the topology.

# 4. Complexity and Simulations

## 4.1 Complexity Analysis

After a modification of the sets by the processing of BGP updates or a change in the IGP distances, BGP needs to find the new best gateway for each destination. Fortunately, since destination points towards shared 1OP sets, this can be achieved by looking for the best gateway of each 1OP set. Therefore, the complexity of an update performed by BGP scales with the number of 1OP sets that have to be gone through, as well as the size of these sets. We will start by studying the number of 1OP sets that needs to be gone through at worse and the resulting complexity, before having a closer look at the impact of the size of the sets.

In practice, we expect the number of 1OP sets to be less than the number of destination prefixes. However, theoretically, the number of sets is limited by two factors: the number of destinations prefixes and the number of border routers (or gateways). More precisely, **at worst**, the number of 1OP sets is equal to the number of prefixes (i.e., each prefix has its own 1OP set), or the number of 1OP sets is equal to every possible combination of gateways (i.e., the destination prefixes cover all the 1OP sets that can be created given the topology).

Consequently, with $N$ the number of 1OP sets, $B$ the number of border routers and $P$ the number of prefixes, at worst $N = min(P, 2^B - 1)$.

Given the number of sets, we can now find an upper bound to the complexity $K$ of the search for the new best gateway for every 1OP set. We know that, if every 1OP set contains every gateway, we need to go through $B$ entries for each set. Thus, if $P < 2^B - 1$, $K < B \times P$ and else, $K < B \times 2^B - 1$. This upper bound cannot be reached. Indeed, each set containing every gateway would result in a single set containing all gateways, shared by every destination. Consequently, to find the complexity of BGP at worst, we need to find the biggest possible number of distinct 1OP sets.

At worst, each possible set containing one gateway exist, as well as each possible set composed of two gateways, and so on until all sets containing $B$ gateways. In other words, every possible set of size $i$ must be gone through, from $i$ going from 1 to $B$. The number of possible sets containing $i$ gateways is equal to $\binom{B}{i}$. Each of these sets contains $i$ gateways, for a total of $i \times \binom{B}{i}$ look-ups. This run-through must, in turn, be done for

every set, i.e., for $i$ from 1 to $B$. Thus, if $P > 2^B - 1$, the total complexity is equal to, according to the binomial theorem:

$$\sum_{i=0}^{B} \binom{B}{i} \times i = \sum_{i=1}^{B} B \times \frac{(B-1)!}{(i-1)! \times (B-i)!} = B \times \sum_{i=1}^{B} \binom{B-1}{i-1} = B \times \sum_{i=0}^{B-1} \binom{B-1}{i} = B \times 2^{B-1} \quad (1)$$

If $P < 2^B - 1$, the number of operations will be slightly lesser than in the previous case. However, as this case is still probable, it is interesting to formalize the complexity of the solution nonetheless. Here, the worst possible case happens if each existing destination prefix needs its own 1OP set, containing the highest possible number of border routers , with all sets staying distinct. This is expressed by the following formula:

$$\sum_{i=k}^{B} \binom{B}{k} \times k, \text{ where } k \text{ is the largest integer such that } \sum_{i=k}^{B} \binom{B}{k} \geq P \quad (2)$$

As seen, we can easily deterministically compute the number of sets in the worst case. However, their size is subject to more complex interactions that depend on iBGP topologies, route dissemination, policies and so on. In practice, the size of the sets plays an important part in this complexity. While the size of the sets itself is one of the two main factors of the complexity of an BGP update, it also impacts the actual number of sets. Before explaining in more details the factors that influence the size of our sets, we will explain how this size turns out to affect the total number of sets.

As mentioned, the sets are shared: consequently, if a new destination uses the same set as another one, there is no need to create a new set. We will refer to this case as a ***merging collision***. The greater the chance of merging collisions, the higher the ratio of destinations per 1OP sets, resulting in less 1OP sets and reduced complexity upon update.

The chance of merging collisions directly depends on the theoretical maximum number of distinct 1OP sets that can be created, i.e., for $B$ border routers, $\sum_{i=1}^{B} \binom{B}{i}$. If this combinatorial space decreases, the number of possible 1OP sets is reduced, resulting in an increased chance of a merging collisions.

As an example, let us imagine a case were every gateway needs to be kept within every one of our 1OP sets. We then have a total number of distinct set equal to $\binom{B}{B} = 1$, resulting in only one possible set containing all the gateways, which would be shared by all the destination prefixes. At the opposite side of the spectrum, if each sets contains only a single gateway, we would create as many sets as there are gateways ($\binom{B}{1} = B$). This number of possible sets peaks when each set contains $\frac{B}{2}$ gateways. $\binom{B}{\frac{B}{2}}$ is indeed the worst case for us, as it is at this value that BGP can create the greater number of distinct 1OP sets. This relation is fairly intuitive. If we randomly pick a set of gateways for our prefixes from a set of 20 gateways, we have a higher chance of picking several times the same couple of gateways than picking the same 10-Tuple.

As a result, in order to maintain a lower number of set (i.e, complexity), we ideally want the average size of our sets to tend towards 2 or $B$, to reduce the number of possible sets and increase the chance of merging collisions. While we can assume that the average set size would initially be $\frac{B}{2}$, in practice, we can expect this mean to naturally be shifted towards a lesser value.

Indeed, even though $\frac{B}{2}$ border routers in average might learn a route towards a prefix, we are only interested in keeping the routes discriminated solely by they IGP cost attribute. The more diverse the policies and previous attributes are, the slimmer the chances of two routes being equal up until their IGP distance attribute becomes,

resulting in smaller 1OP sets and an increased chance of merging collisions. We refer to the diversity of BGP attributes as *policy spreading area* (*ps*). For example, a policy spreading area of 10 means that the vector created by the BGP attributes of a route up until the IGP cost can take 10 possible different values. Consequently, there is a chance of $\frac{1}{10}$ for two routes to be discriminated only by their IGP distances and to be kept within the same 1OP set. We could thus expect the number of 1OP set to quickly decrease as the policy spreading area increases. In summary, while the number of border routers learning each destination is linked to the maximum theoretical amount of distinct 1OP sets, it is the value of the policy spreading area which will allow to reduce the combinatorial space further by filtering out routes.

In addition to the policy spreading, the size of 1OP sets is also reduced by the loss of route knowledge occurring within iBGP topologies mentioned in section 2.3. While some solutions allow for every router to know the full extent of routes learned towards a destination, most architectures may suffer from this loss of information. Of course, in order to maintain the 1OP property, we do not want our sets to be reduced too much. But in a large enough network, we can assume that, even if the average size of a 1OP sets is less than $\frac{B}{2}$ originally, these phenomenons would reduce the mean size without endangering the 1OP property of the sets.

Given the policy spreading, we can predict a theoretical threshold of the number of 1OP sets. Indeed, for a policy spreading *ps*, and assuming that the attributes of the routes are distributed evenly, we should theoretically have at most $\frac{B}{ps}$ routes tied at the IGP cost attribute. Consequently, when enough destinations are considered, all possibles distinct sets of size 2 to $\frac{B}{ps}$ can be created, translating to the formula: $\sum_{i=2}^{\frac{B}{ps}} \binom{B}{i}$ (3)

However, in practice, some sets containing more than $\frac{B}{ps}$ may be created. While these sets should represent a small portion of the total number of 1OP sets, they may have a strong impact on the complexity of BGP. Indeed, given the size and prevalence of these outliers, the probability of a merging collision between them is low as the associated combinatorial space is quite vast. Consequently, it is possible for each outliers to generate a new distinct 1OP sets. The number of outliers is directly tied to the total number of destinations, as more of them can appear if more destinations are considered. Therefore, while formula (3) gives a fairly good approximation of the number of sets, estimating precisely the complexity of BGP requires to take more parameters into account, such as the number of prefixes, different distributions regarding the number of routers advertising a route, non-uniform policy spreading distributions...

To this end, a simulation tool, LenSim[2], was coded in python, allowing us to finely tune a vast array of parameters in order to better assess the complexity of OPTIC.

## 4.2 Simulation Results

LenSim simulates how OPTIC would behave when run on a BGP speaker upon the reception of various advertisements from its peers. Let $B$ be the number of border routers and *ps* the aforementioned policy spreading. For every destinations, LenSim will randomly chose a number $AR$ of Advertising Routers between 2 and $B$, representing the number of border routers having learned and advertising the route. LenSim will then generate a total of $AR$ advertisements, that associated with a random number between 1 and *ps* to represent the BGP attributes of the route. We refer to this number as the ***attractiveness*** of the route.

More precisely, the attractiveness of the route only represent its local-pref and as-path

[2]https://gitlab.unistra.fr/jrluttringer/lensim

attribute. Therefore, If two routes $r$ and $r'$ share the same attractiveness, $r =_I r'$. Once these advertisements are generated, they are processed as they would be processed by a router running OPTIC. When all the destinations have been processed, the number of resulting 1OP sets is extracted. LenSim allows us to tune the number of border routers, the number of prefixes, the number of advertising routers, the policy spreading area as well as the distribution of the randomly picked value, the number of advertising routers and various other parameters. We will here only use uniform distributions and vary the number of advertising routers.

LenSim does not take into account the internal topology information and cannot thus simulate the construction of minimal 1OP sets. We will limit our study here to the construction of 2R and 2R-optimized set (supposing gateway independence).

Our main goal is to study the impact of the policy spreading and the number of advertising routers on BGP, as it is one of the main factor that might limit the number of 1OP sets created by out solution.

We consider here 100 border routers and 800000 prefixes. To gradually reflect more and more intricate policies and a broader spectrum of BGP attributes, we increase the policy spreading area from 1 to 1024 following the powers of 2. Regarding the number of advertising routers, we consider two different values. First, we will study the evolution of the number of 1OP sets if each prefixes is learned by between 2 and 100 border routers ($2 \leq AR \leq 100$). Second, we will study the number of 1OP sets if each prefixes is learned by between 2 and $\frac{B}{5}$ routers ($2 \leq AR \leq 20$). The advertising routers for each prefix are selected randomly from all the border routers. As the number of advertising routers impacts the maximum possible number of distinct 1OP sets, we expect this value to have a great incidence on the complexity of OPTIC. Finally, we compare our results to PIC, when PIC creates the smallest sets possible, i.e., two gateways per sets. While this is not the best case for PIC in terms of resiliency, it is the best in terms of complexity. Furthermore, as most ISPs do not implement BGP multi-path policies, PIC would, most of the time, only store the two best routes in its ECMP sets [6]. It is important to note that we do not for OPTIC's number of sets to be equal to PIC's number of path lists. Since OPTIC provide far more guarantees than PIC and is thus bound to be more complex, we simply want to check if OPTIC's number of sets is is the same magnitude as PIC's number of PLs. In addition, these simulation were favorable to PIC, given that we assume that PIC only keeps the smallest number of route possible, while it is disadvantageous to OPTIC, as the number of advertising routers is a lot higher than what we would expect to see in practice. While the experiments were done 30 times in order to extract a mean value with its associated confidence intervals, these latter are not broad enough to appear on the resulting graphs.

Figure 19 illustrates the results of our simulations regarding the number of 1OP sets. We use a double logarithmic scale to show our data. The top-most line in orange represents the number of entries in a flat FIB, which is always equal to the number of prefixes. Next, the blue lines with round markers show the number of 1OP sets created by OPTIC when the number of advertising routers is between 2 and 20 for each route. The red and pink lines with square markers show the number of 1OP sets created by OPTIC the number of advertising routers is between 2 and 100 for each route. Finally, the black line shows the number of BGP PLs created by PIC, if PIC only keeps the two best gateways, while the dashed grey line shows the theoretical approximation given by formula (3). These numbers are given depending on the policy spreading area. Figure 20 illustrated the average size of 1OP sets for the same simulations.

From a global point of view, we can see that OPTIC, particularly its optimized variants, performs quite well, especially given the disadvantageous nature of the simulation.

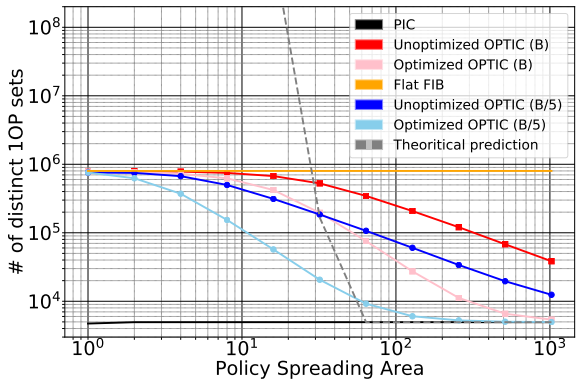



Figure 19: Average number of 1OP sets depending on the number of possible combinations of BGP attributes.
The number of BR advertising each prefixes has a strong influence over the number of distinct 1OP sets. While OPTIC fares better than the theoretical prediction at first, the bigger outlier sets increase the number of distinct 1OP sets.

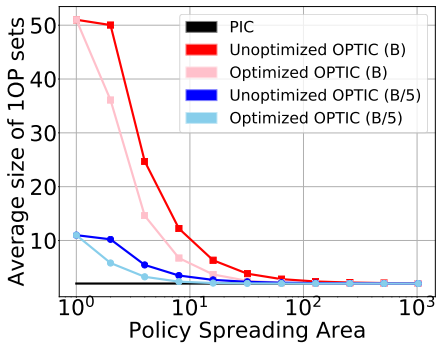



Figure 20: Average size of 1OP sets depending on the number of possible combinations of BGP attributes.
The optimized 2R sets allow the size of 1OP sets to decrease more quickly. The policy spreading area allows to limit the size of the sets.

Indeed, while OPTIC's number of sets seems high at first, it still manages to reach PIC's number of sets magnitude, or be equal to PIC's number of sets for a high enough policy spreading, even when we consider a situation highly favorable to PIC.

As mentioned, a higher policy spreading limits the chance of routes sharing the same attractiveness, i.e, to be tied at the IGP cost rule. This in turns reduces the size of our sets and the number of possible distinct sets. As can be seen on Figure 20, the size of 1OP sets decreases quickly, going from 50 (with 100 AR maximum ) and 10 (with 10 AR maximum) to 2 for a policy spreading of 128 and 32 respectively. We could expect the number of distinct 1OP sets to reach its minimum once the policy spreading reaches the average number of advertising routers. Indeed, since only $\frac{AR}{ps}$ gateways should, on average, share the same best attractiveness, a policy spreading equal to the number of AR lead to sets containing $\frac{AR}{AR} = 1$ gateway, which we increase to 2 so that our set are 2R, giving $\binom{B}{2}$ possible sets.

However, we can see that this minimum (about 5000 sets) is not reached until later, for a policy spreading greater than 512 or greater than 1024 depending on the number of advertising routers, meaning that some disruption causes the number of total distinct 1OP sets to be higher than expected. Indeed, While our theoretical prediction drops quickly, reaching a number of sets equal to PIC for a policy spreading of 100, OPTIC exhibits a far more gentle slope. However, one can see that the average size of 1OP sets yet seems to have reached its minimum for $ps = 100$.

This strong difference from our theoretical approach comes from the bigger outlier sets that are generated due to the random nature of our simulation. For 800000 prefixes, about 7000 routes are still advertised by at least 3 border routers with an equal attractiveness. There are no merging collisions between these outliers: each of these prefixes generates its own distinct 1OP set, explaining both the difference between OPTIC and our theoretical approach as well as the slower than expected slope. However, optimized 2R sets bring the size of some of these outliers back to 2.

OPTIC, optimized or not, creates almost as many 2R sets as there are prefixes when the policy spreading area is small. Indeed, as routes have a very high chance to share the same attractiveness. Due to this, almost all the advertising routers have to be kept within

the associated IGP rounded sets, resulting in large sets and thus a low chance of merging collision. However, using optimized 2R sets allows OPTIC to reach an almost minimal number of sets far more quickly (for a policy spreading of 256 and 1024 depending on the number of advertising routers). As can be seen in Figure 19, this minimal value is only reached by OPTIC when using optimized 2R. While both variants reach the minimal possible average size of 1OP sets, the fact that unoptimized variants of OPTIC cannot always allow a set to be reduced to two elements greatly increases the number of outliers. Thanks to the gateway independence hypothesis, the optimized version of OPTIC can filter some of these outliers, resulting in a reduced number of distinct 1OP sets.

While OPTIC does not manage to reach the complexity of PIC with $B$ advertising routers , this case is an extreme side of the spectrum; we can expect the other case, i.e., $\frac{B}{5}$ advertising routers at most, to better reflect real-world scenarios. In the former case, the number of sets generated by OPTIC is slightly higher than PIC's for a policy spreading of 256 (11192 for the optimized version and 50000 for PIC). However, in the latter case, optimized variants of OPTIC manage to reach PIC's complexity for a policy spreading of 256 creating 5000 distinct set.

Even though OPTIC may not always reach the number of sets created by PIC, it is important to note that the simulation is clearly not to OPTIC's advantage, as we do not benefit from the effects of iBGP topology, filters, polices, or realistic BGP attributes. In particular, we can expect the number of advertising routers for each prefix to be far lesser in practice, which would greatly improve OPTIC's results. In practice, preliminary tests on more realistic data tend to show that real-world parameters are indeed more advantageous to OPTIC than the simulations done here. Besides, even though it is slightly more complex, OPTIC, on the contrary to PIC, guarantees the protection of each destination upon intra-domain failure, and quick re-convergence to the new optimal path with no transitionnary states.

# Chapter 5 Conclusion

## 1. Contributions

In this thesis, we studied primarily the effect of intra-domain failures on BGP convergence and inter-domain routing. After having studied the various protocols coming into play during traffic forwarding, we studied the various existing mechanisms that allow fast recovery upon failure. In particular, we studied PIC, using a personalized formal model, and analyzed the current weak points of these methods. We have shown that, depending on the topology, guaranteeing fast BGP recovery after an internal failure may be complex, and guaranteeing an optimal fast recovery could be even more complicated due to the delicate intricacy between inter and intra-domain routing protocols.

We then introduced several new constructions that, when used, allow to creates sets of BGP NH that ensures quick and optimal recovery of BGP upon an intra-domain failure. We showed how OPTIC constructs and maintains such a kind of sets upon the reception of BGP updates. We then formally prove that the proposed constructs and algorithms are correct. While we first relied on a fairly restrictive hypothesis to ease the introduction of these new constructions and algorithms, we lifted this hypothesis later on and proposed some optimizations to reduce the complexity of the operations that OPTIC must run upon an intra-domain change or a BGP update. Finally, we conducted a theoretical analysis on the complexity of OPTIC and ran several simulation to study the complexity of OPTIC and analyze the impact of BGP policies on the complexity of our solution. While OPTIC's complexity is above the complexity of existing solutions, we have shown that the main advantage of OPTIC is to guarantee an optimal quick recovery of the transiting traffic upon an intra-domain failure. In addition, real-world conditions might work to OPTIC's advantage, and allows it to approach, if not be equal to, the complexity of existing solutions.

## 2. Perspectives

Several options can be explored to extend OPTIC. Different ways to compute 1OP set can be explored, to fully browse the spectrum of the compromise between the minimality of the sets and their computation. These solutions could then be tested on real-world data while taking into account the underlying iBGP topology. Indeed, while we know that the iBGP policies and topology plays an important role in the creation of our sets, we could not analyze this interaction further using real data. Of course, one of the most interesting perspective is to actually implement OPTIC on a real router, or a routing suite. Many optimizations could be done and challenges may arise when considering a concrete implementation of our algorithms. One can also even imagine a closer interaction between OPTIC and IGPs, in order to slightly modify, when viable, the weight of some links to allow the use of optimized 1OP sets. Quite a lot of possible tweaks and adjustments can be done to improve the implementation of OPTIC, as the use and tuning of timers to destroy unused 1OP set depending on the frequency of failures, the use of various optimized data structures to improve the update time of OPTIC or improving the interactions between OPTIC and IPFRR mechanisms as well as Layer 2

fast rerouting mechanisms.

To conclude on a more personal note, this thesis allowed me to become more familiar with various complex and fascinating subjects. While the subject slightly deviated from what we had in mind, to become a far more complex and theoretical subject than we expected, this departure turned out to be a blessing in disguise. Indeed, the new problems that arose were particularly fascinating and motivating. They allowed me to get a better grasp on algorithmic, graph theory and combinatorics, which, being the cornerstones of computer science and networking, I wish I was more familiar with. In addition, I gained a deeper understanding of various technical subjects such as routing protocols and the interactions between them, making this subject a very interesting traversal of a wide and varied array of compelling challenges.

# Appendix A
# Theorems, Lemmas, Definitions and Properties

- **Representation of BGP routes**
  We consider BGP routes $r$ as a triplet $(r_d, r_g, r_a)$ with $r_d$ the destination, $r_g$ the gateway (or BGP NH), and $r_a$ the attributes of the route. We simplify $r_a$ as (Local-pref, AS-path, IGP cost, Tie-break), or (LP,AS,I,TB).

- **Personalized Operators**
  $\prec, \succ$ and $=$ are operators used to lexicographically compare two vectors. $\prec_x, \succ_x$ and $=_x$ are used to lexicographically compare two vectors up until the $x^{th}$ element (not included).

- **Definition 1 (1-Optimal-Protecting (1OP) Sets)**
  A set containing, for each destination prefix, the best gateway before internal failure and all the possible best gateways after any internal failure.

- **Definition 2 (Minimal 1OP Sets)**
  A set containing, for each destination prefix, the best gateway before internal failure and all the possible best gateways after any internal failure, but no other gateway. Non minimal 1OP sets are referred to as **relaxed 1OP sets**.

- **Definition 3 (IGP rounded sets)**
  A set containing a BGP NH $b$ and all the BGP NHs $b'$ that share the same BGP attributes up until their IGP distances I $(\forall b', b =_I b')$.

- **Lemma 1 (Order persistence upon internal failure)**
  The order given by the operators $\prec_I$, $=_I$ and $\succ_I$ is not affected by an intra-domain failure.

- **Definition 4 (2-Rounded (2R) sets)**
  A set composed of the two best IGP rounded sets, or only the best IGP rounded set if it contains at least two gateways.

- **Theorem 1 (2R sets property)**
  2R sets are 1OP if the network is biconnected.

- **Property 1 (2R UPDATE persistence)**
  OPTIC maintains or creates 2R sets upon the reception of a BGP `UPDATE` message

- **Property 2 (2R WITHDRAW persistence)**
  OPTIC maintains or creates 2R sets upon the reception of a BGP `WITHDRAW` messages

- **Theorem 2 (2R OPTIC)**
  OPTIC creates and maintains 2R sets

- **Definition 5 (P-Rounded sets)**
  The union of $P$ rounded sets, containing at least $p$ gateways, where $p$ is the number of gateways needed to ensure the protection of the destination $d$.

# Appendix B
# OPTIC and FRR mechanisms

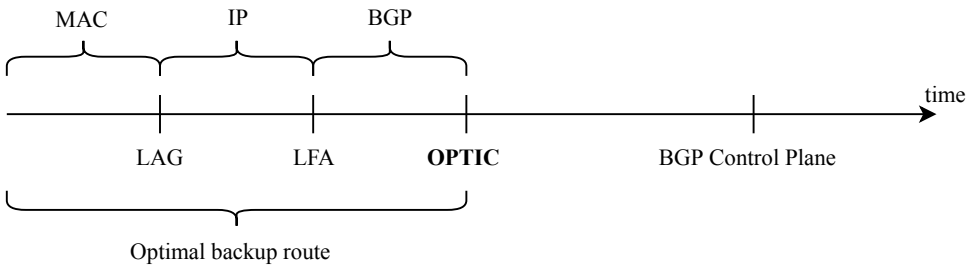



Figure 21: OPTIC with others FRR mechanisms

OPTIC can be seen at the end of the spectrum of FRR mechanism that allow quick recovery upon intra-domain failure. Indeed, an intra-domain failure may not change the IGP distance towards the gateways in a significant way, i.e., the best gateway towards the destination $r_d$ might not change depending on the failure.

In this case, finding the new intra-domain route towards the best gateway is enough: there is no need to wait for OPTIC to converge to ensure the optimality of the transiting traffic. Thus, ensure the protection and optimality of the transiting traffic can be seen as an incremental process, as shown by Figure 21. If LAG, at the layer 2, is enough to restore the connectivity in an optimal fashion, updating OPTIC (if needed) can be done in the background at a further point. The same reasoning can be done if an LFA or similar Layer FRR mechanism allows for an optimal recovery. If quicker, simpler methods failed to restore the optimality of the connectivity of the transiting traffic, OPTIC must be used. However, since we showed that OPTIC is always able to find the new best route quickly, there is no need to wait for BGP to converge naturally before ensure that the transiting traffic is following the optimal route again.

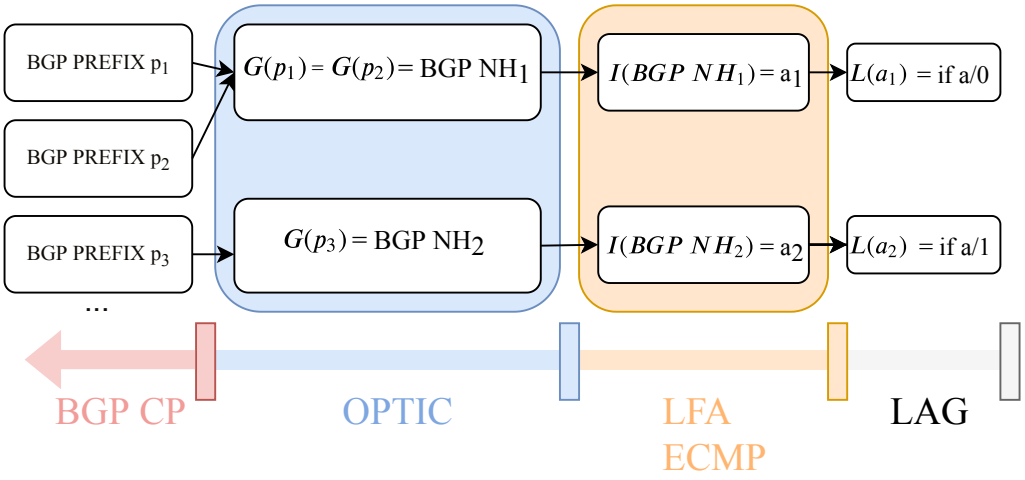



Figure 22: Updating the HFIB with several FRR mechanisms

This incremental way of dealing with a failure can be seen as updating each level of an HFIB, starting from the leaves and going to the roots, as shown in Figure 22. Starting with the leaves ($L$ function), while the optimality and protection are not achieve, updating the level above quickly (thanks to the associated mechanisms)is necessary. Once these conditions are reached, it is not as critical to update the remaining levels.

# Appendix C
# Managing 1OP sets

Firstly, as mentioned in this thesis,1OP sets are shared across destinations to achieve a better update time. Indeed, if several destinations share the same best gateway as well as the same best gateways upon failure, there is no need to maintain separate 1OP sets for these destinations.

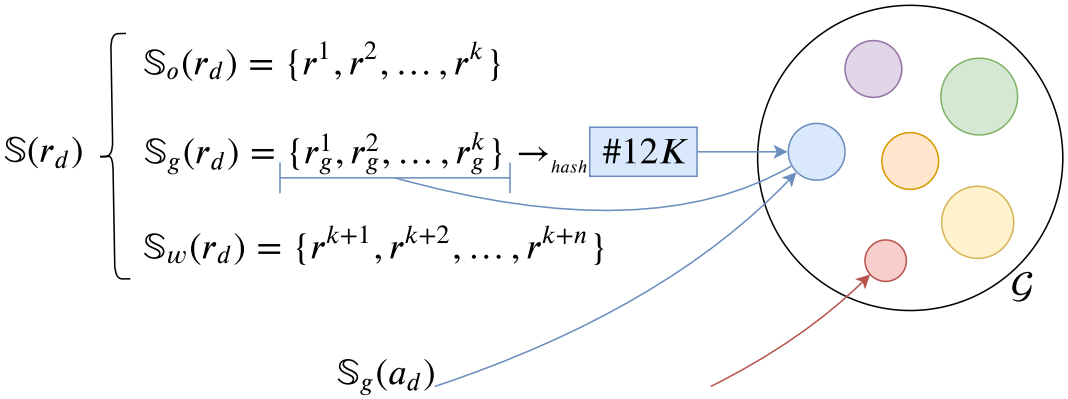



Figure 23: OPTIC shared 1OP sets

Therefore, we can imagine OPTIC maintaining a large structure $\mathcal{G}$ containing the existing 1OP sets, illustrated as colored circles in Figure 23. Upon applying a hash function on the computed 1OP set of a destination, we can check if the same 1OP set was already created for another destination. For example, in Figure 23, the application of an hash function on the 1OP set $\{r_g^1, r_g^2, ..., r_g^k\}$ gives us $\#12K$, which can be seen as an index referencing a 1OP set in $\mathcal{G}$. As the 1OP set composed of these gateways was already created for $a_d$, we do not need to recreate it. Instead, we simply state that $r_d$ and $a_d$ points towards the same 1OP set, i.e., the blue 1OP in the Figure. Upon changes, updating the blue 1OP will update the best gateway for both $r_d$ and $a_d$. This allow to ensure that the number of 1OP set will not be larger than the number of destinations.

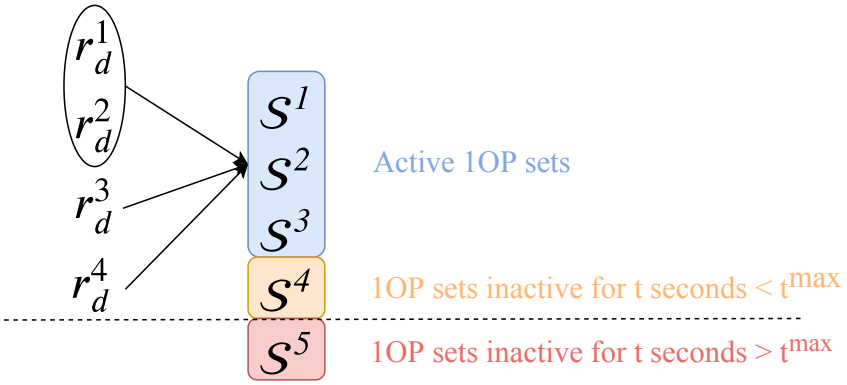



Figure 24: Destroying sets while mitigating over-reactions

While creating and destroying sets may be costly, we can imagine a system where these operations do not need to be done immediately, to ensure that the sets that are destroyed are obsolete. Indeed, let us imagine a destination $r_d$ with its associated 1OP set $S^4$. Upon the reception of a withdraw message, one of the gateway of $S^4$ becomes unreachable. Let $S^1$, the new 1OP set of $r_d$, be an already existing 1OP set. Thus, we make $r_d$ point toward $S^1$ and we destroy $S^4$. If the failure was temporary and the gateway becomes reachable again, we need to recreate $S^4$. To prevent these over-reactions, we can associate a timer to unused sets (i.e., set towards which no destination point), and destroy the sets whose timers are expired when the additional workload does not prevent the router from handling traffic. This is shown by Figure 24. The sets are considered as expired if they are inactive for more than $t^{max}$ seconds. The sets that have been inactive for a lesser period are kept in case they become active again. The expired set can be destroyed periodically by a garbage collector-type of mechanism when the workload of the router allows it.

# Appendix D
# OPTIC States Diagram

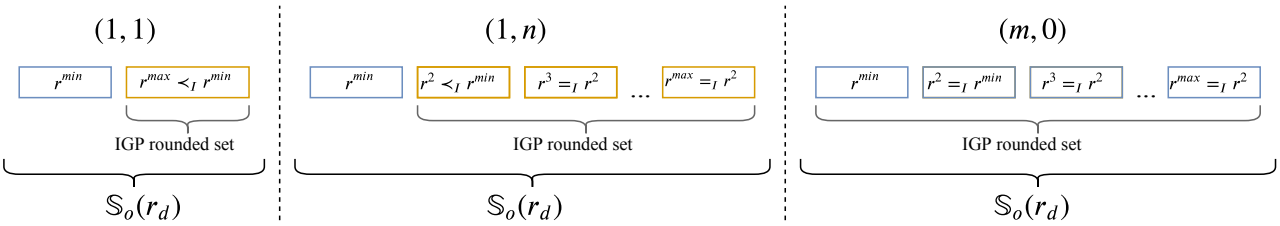



Figure 25: Viable sets for OPTIC. Each of these sets show one of the possible decomposition of 2R set in one or two distinct IGP rounded sets.

While OPTIC's proofs were formals, we can understand OPTIC at a more high-level perspective to get a better intuition regarding its inner workings. To better understand OPTIC, it is useful to imagine $\mathbb{S}_g(r_d)$ (the 1OP set of gateways) as a union of subsets $\mathcal{G}_1, \mathcal{G}_2$, such that $\mathcal{G}_1$ and $\mathcal{G}_2$ are composed of 1 to n gateways each. The definition of 2R sets allows us to list explicitly the type of couples of subsets $(\mathcal{G}_1, \mathcal{G}_2)$ that creates 2R sets. Indeed, as explained during the description of 2R sets, only a few types of couples can result from this description. These couples are illustrated in Figure 25. We will represent these subsets as a couple of integers representing the cardinal of their subset $\mathcal{G}_1$ and $\mathcal{G}_2$.

- $(m, 0)$: $|\mathcal{G}_1| = m$ and $\mathcal{G}_2 = \varnothing$ with $m \geq 2$ (one IGP rounded set containing at least two elements).
- $(1, n)$: $|\mathcal{G}_1| = 1$ and $|\mathcal{G}_2| = n$ with $n \geq 2 : \mathcal{G}_1$ contains the best route. If this route fails, since we do not predict how the failure will affect the `IGP cost` attribute of back up routes, $\mathcal{G}_2$ is IGP rounded. We have the two best IGP rounded sets.
- $(1, 1)$ $|\mathcal{G}_1| = 1$ and $|\mathcal{G}_2| = 1$: $\mathcal{G}_1$ contains the best route. There is no two back up routes $r^1$ and $r^2$ such that $r^1 =_I r^2$, thus $\mathcal{G}_2$ does not need to contain more that 1 route, as an intra-domain failure would not change this equality. We have the two best IGP rounded sets.

Since these states respect the definition of 2R sets, we want OPTIC to transition between these states. We will later refer to those couples as viable states. When we refer to $\mathbb{S}_g(r_d)$ as being IGP rounded, we refer to the $(m, 0)$ state.

Note that the viable state $(1, 1)$ is a sub-case of $(1, n)$ where $n = 1$. In other terms, $\mathcal{G}_2$ is also an IGP rounded subset containing only one element. The main purpose of displaying this state was to showcase the fact that an IGP rounded subset can contain only one element. However, to simplify the following explanation, we will merge these two states. We are thus left with only two viable states: $(1, n)$ and $(m, 0)$, with $n \geq 1$ and $m \geq 2$.

We can now look back at the Listing 4.1, and showcase how we ensure that our algorithm indeed transitions between viable states. The comments in the code are again going to be used as labels to link the transitions between sets and the explanations to the corresponding parts of our algorithm. We will list all the transitions that may occur

in OPTIC. We will see that upon a modification, OPTIC always brings the 2R set back to one of the viable state, which are 2R by construction, upon the processing of a route $r$.

- If $r$ is `non-optimal`, and two gateways are already stored in $\mathcal{G}$, we are either in the $(m, 0)$ or $(1, n)$ state. If we were to add $r$ to our set, (in the underlined subset), due to the fact that $r$ is worst than the existing routes, these states would become $(m, \underline{1})$, or $(1, n, \underline{1})$. As no state are 2R (viable), we put $r$, the worst route, aside in $\mathbb{S}_w$, to get back to the viable states $(m, 0)$ or $(1, n)$.
- If $r$ is the new `absolute best route`, we can be in all possible cases evoked above. After adding $r$, our subsets become $(\underline{1}, m)$ or $(\underline{1}, 1, n)$. We can see that in the first case (i.e., when $\mathcal{G}$ is IGP rounded), we can simply add the new route and still be in a viable state, while in the other case, we should only keep the two best routes to get back to a viable type of set, i.e. $(\underline{1}, 1)$. These choices can be seen in the Listing.
- If $\mathcal{G}$ is IGP rounded, and $r$ is neither worse than the current worst route nor is it better than the current best route, by adding $r$ to our set, we will move from $(m, 0)$ to $(\underline{m}, 0)$, which remains a viable state: we can add the new route (`equivalent route`).
- The remaining case is that $r \prec_I r_{max}$ and the set is not IGP rounded. It is useful to decompose this, as in the Listing, in two subcases $r \prec_I r_{max}$ and $r =_I r_{max}$.

  In the former case, since $\mathcal{G}$ is not IGP rounded, the only possible state left at this point of the algorithm is $(1, n)$, resulting in $(1, \underline{1}, n)$ after adding the new route. We thus need to keep the two best routes to stay in a viable state $(1, \underline{1})$.

  In the later subcase, the only possible state is $(1, n)$. After adding the new route, this state becomes $(1, \underline{n})$, meaning that we can add the new route while keeping our set in a viable state.

We can show, in the same fashion, that Listing 4.2 also allows OPTIC to go between viable sets. We ignore the `Nothing to be done` case, as the route being withdrawn was not used by OPTIC to begin with.

- In the `Still 2R` case, we either stay in the $(1, m)$ or $(m, 0)$ states or go from $(1, m)$ to $(m, 0)$, as we remove a route from a set that contained at least three routes.
- In the `Add necessary routes` case, we go from the $(1, 0)$ case to either $(m, 0)$ or $(1, n)$.

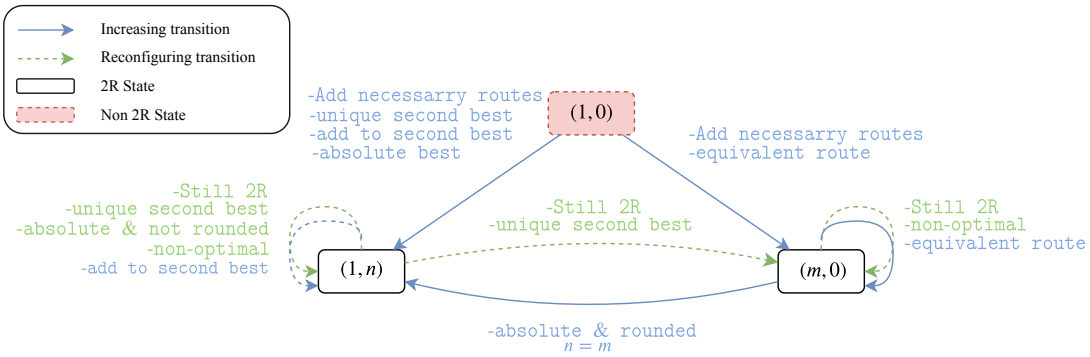



Figure 26: State transitions of OPTIC sets. This diagram shows the transition performed by our algorithm from one viable state to another. While blue transitions only add routes, green dashed transitions may also remove routes from our set. Each modification keeps the set in a 2R state

To give a visual intuition of how our algorithm switches between these states, the transitions were be summed up in the state diagram shown in Figure 26. The blue transitions are *increasing*, meaning that we add a new route to our set. The green dashed transitions are *reconfiguring*: we discard routes from our set. Our proofs, we ensured that the increasing transitions maintain an efficient 2R set by adding necessary routes, while reconfiguring transitions main an efficient 2R set by adding necessary routes as well as removing unnecessary routes. While we did not talk extensively about the $(1, 0)$ state, this state is only temporary and can exist only if a single gateway to a destination $d$ is known, i.e., when the routes are not sufficient to create 2R sets. We can see that OPTIC goes between viable states that are 2R sets, and we showed thanks to Property 1 and 2 that the transition between these state maintains the 2R property, ensuring that OPTIC creates 1OP sets.

# Glossary

**1OP** — 1-Optimal-Protecting — A 1OP set ensure the protection of a destination upon failure and the optimality of the backup route used **(Contribution)**. 29, 30, 32, 33, 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, 49, 57, 59

**2R** — 2-Rounded — A 2R set is a 1OP set composed of the union of the best IGP rounded set of routes, containing 2 routes at least**(Contribution)**. 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 45, 46, 57, 58, 59

**AR** — Advertising Router — Router advertising an external destination to its peers. 44

**AS** — Autonomous System — A set of routers which are under the same administrative authority. 9, 10, 11, 12, 13, 14, 15, 17, 20, 21, 22, 24

**ASBR** — Autonomous System Border Router — A router at the border of an AS, which usually links the AS to a neighboring AS. 9

**attractiveness** — Number representing the vector of BGP attributes of a BGP route.two routes $r$ and $r'$ sharethe same attractiveness if $r =_I r'$. 44

**BGP** — Border Gateway Protocol — The *de facto* inter-domain routing protocol. 2, 5, 9, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 29, 30, 31, 33, 34, 37, 39, 42, 43, 44, 45, 47, 49

**eBGP** — Exterior Border Gateway Protocol — A BGP mode used between different autonomous systems. 9, 11, 12, 22

**ECMP** — Equal Cost Multi Path — A routing strategy allowing data to follow multiple best paths having the same weight. 8, 19, 22, 23, 24, 26, 29

**FIB** — Forwarding Information Base — A table containing the resolved best next-hops towards each destination. 5, 13, 15, 22, 23

**FRR** — Fast ReRoute — Mechanisms allowing a quick recovery upon failure thanks to precomputed routes. 7, 9, 29

**HFIB** — Hierarchical Forwarding Information Base — A FIB that uses a hierarchical architecture to greatly reduce its update time. 23, 24

**iBGP** Interior Border Gateway Protocol — A BGP mode used to exchange BGP routes between BGP speakers within the same AS.. 2, 9, 11, 15, 17, 49

**IGP** Interior Gateway Protocol — A routing protocol used within a domain. 2, 5, 6, 8, 9, 11, 12, 13, 16, 18, 19, 20, 21, 22, 23, 24, 26, 30, 31, 32, 35, 37, 38, 39, 40, 41, 42, 43, 44, 46, 49, 57, 58

**ISP** Internet Service Provider — An actor that provides services for accessing the Internet. 2, 17

**LFA** Loop Free Alternate — A FRR mechanism allowing a node to use a neighbor to reach a backup path upon failure. 8, 9

**merging collisions** Event referencing to two destinations having the same 1OP set, allowing them to share said set. 43

**NH** Next Hop — The next optimal router a packet must go through to reach a destination. 20, 23, 24, 29, 49

**OPTIC** Optimal Protection Technique for Intradomain Convergence — A method that allows BGP to recover optimally from an intra-domain failure **(Contribution)**. 2, 26, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 44, 45, 46, 47, 49, 57, 58, 59

**OSPF** Open Shortest Path First — A Link-state routing protocol. 7

**PIC** Prefix Independent Convergence — A method allowing quicker BGP recovery, independent from the number of prefixes stored. 22, 23, 24, 25, 26, 29, 39, 45, 47, 49

**PL** Path List — A structure used by PIC to store ECMP routes. 24, 45

**ps** Policy Spreading — The diversity of the routing policies using. 44

**R-BGP** Resilient Border Gateway Protocol — A method allowing to isolate the data from remote BGP failure. 17, 18

**relaxed 1OP sets** 1OP sets that are not minimal. 30

**RIB** Routing Information Base — A table that contains all the routes to each available destination. 5, 11, 22, 24

**RIP** Routing Information Protocol —A distance-vector protocol. 6

**RLFA** Remote Loop Free Alternate — A LFA which is not a direct neighbor. 8, 9

**SLA** Service Level Agreements — A commitment between an ISP and a client regarding quality of service. 2

# Nomenclature

| | |
|---|---|
| $\alpha$ | Intra-domain path. More precisely, $\alpha_x$ is the best intra domain path towards $x$. May also be used to reference the cost of the path |
| $\beta$ | Inter-domain related attributes of a BGP route's vector of attribute |
| $\mathbb{S}$ | Set containing three sets $\mathbb{S}_o, \mathbb{S}_g, \mathbb{S}_w$. We use $\mathbb{S}(d)$ to refer to the set $\mathbb{S}$ associated with the destination $d$. |
| $\mathbb{S}_g$ or $\mathcal{G}$ | Set of gateways extracted from the routes within $\mathbb{S}_o$. In particular, for OPTIC, the set of routes that ensures optimality and protection. |
| $\mathbb{S}_o$ or $\mathcal{O}$ | Set of tied optimal routes towards a destination. In particular, for OPTIC, the set of routes that ensures optimality and protection. |
| $\mathbb{S}_w$ or $\mathcal{W}$ | Set of routes that are not optimal. These routes are kept in case optimal routes are withdrawn. |
| $\prec, =, \succ$ | Path comparison operators. |
| $\prec_x, =_x, \succ_x$ | Lexicographical comparison up until the x element (not included). In particular, we use $\prec_I, =_I, \succ_I$ to compare BGP routes up until their IGP distance (i.e., only taking into account the local-pref and the AS-path attributes). |
| $PORT(d)$ | Function computing the outgoing interface for a destination. Composition of $P$ (to get the prefix), $G$ (to get the gateway), $IH$ (to get the IGP next hop), and $L$ (to get the interface). |
| $r$ | BGP route, considered as a triplet $(r_a, r_d, r_g)$. |
| $r^{best}$ | Best BGP route within $\mathbb{S}_o$. |
| $r^{nd}$ | Second best BGP route within $\mathbb{S}_o$. |
| $r^{new}$ | Newly received route. |
| $r^{worst}$ | Worst BGP route within $\mathbb{S}_o$. |
| $r_a$ | Vector of attributes of a BGP routes, simplified to (local-pref, as-path, IGP cost, tie-break), or (LP, AS, I, TB). Can be decomposed as $(\beta \circ \alpha)$ |
| $r_d$ | Destination of the BGP route $r$. |
| $r_g$ | Gateway of the BGP route $r$. |
| **max** | Last vector of a set when ordered lexicographically. |
| **min** | First vector of a set when ordered lexicographically. |

# Bibliography


[1] A Atlas and A Zinin. Basic Specification for IP Fast Reroute: Loop-Free Alternates. RFC 5286, RFC Editor, 2008.

[2] T Bates, E. Chen, and R. Chandra. BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP). Technical report, RFC Editor, 2006.

[3] S Bryant, C Filsfils, S Previdi, M Shand, and N So. Remote Loop-Free Alternate (LFA) Fast Reroute (FRR). RFC 7490, RFC Editor, 2015.

[4] S Bryant, S Previdi, and M Shand. A Framework for IP and MPLS Fast Reroute Using Not-Via Addresses. RFC 6981, RFC Editor, 2013.

[5] G Enyedi, A Csaszar, A Atlas, C Bowers, and A Gopalan. An Algorithm for Computing IP/LDP Fast Reroute Using Maximally Redundant Trees (MRT-FRR). RFC 7811, RFC Editor, 2016.

[6] Clarence Filsfils, Pradosh Mohapatra, John Bettink, Pranav Dharwadkar, Peter De Vriendt, Yuri Tsier, Virginie Van Den Schrieck, Olivier Bonaventure, and Pierre Francois. BGP prefix independent convergence (PIC) technical report. (November 2007):1–14, 2007.

[7] Clarence Filsfils, Stefano Previdi, Bruno Decraene, Stephane Litkowski, and Rob Shakir. Segment Routing Architecture. Internet-Draft draft-ietf-spring-segment-routing-11, Internet Engineering Task Force, 2017.

[8] Pierre Francois, Clarence Filsfils, John Evans, and Olivier Bonaventure. Achieving sub-second IGP convergence in large IP networks. *ACM SIGCOMM Computer Communication Review*, 35(3):35, 2005.

[9] L Gao and J Rexford. Stable Internet Routing Without Global Coordination. *IEEE/ACM Transactions on Networking*, pages 681–692, 2000.

[10] Timothy G Griffin, F Bruce Shepherd, and Gordon Wilfong. The Stable Paths Problem and Interdomain Routing. *IEEE Transactions on Networking*, 2002.

[11] Timothy G. Griffin and Gordon Wilfong. On the correctness of IBGP configuration. *ACM SIGCOMM Computer Communication Review*, 32(4):17, 2004.

[12] Alexander Gurney and Timothy G. Griffin. Metarouting and network optimization. *2006 IEEE Conference on Information Sciences and Systems, CISS 2006 - Proceedings*, page 674, 2007.

[13] Nikola Gvozdiev, Brad Karp, and Mark Handley. LOUP : The Principles and Practice of Intra-Domain Route Dissemination. *Usenix Nsdi*, pages 413–426, 2013.

[14] Thomas Holterbach, Edgar Costa Molero, Alberto Dainotti, Stefano Vissicchio, and Laurent Vanbever. Blink : Fast Connectivity Recovery Entirely in the Data Plane. *Nsdi*, 2019.

[15] Thomas Holterbach, Stefano Vissicchio, Alberto Dainotti, and Laurent Vanbever. SWIFT: Predictive Fast Reroute. *Sigcomm*, pages 460–473, 2017.

[16] Elisa Jasinska, Nick Hilliard, Robert Raszuk, and Niels Bakker. Internet Exchange BGP Route Server. Technical Report 7947, 2016.

[17] Nate Kushman, Srikanth Kandula, Dina Katabi, and Bruce M Maggs. R-BGP: Staying Connected In a Connected World. *Usenix Nsdi*, page 14, 2007.

[18] Craig Labovitz, Abha Ahuja, Abhijit Bose, Farnam Jahanian, Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed Internet routing convergence. *ACM SIGCOMM Computer Communication Review*, 30(4):175–187, 2000.

[19] Stephane Litkowski, Ahmed Bashandy, Clarence Filsfils, Bruno Decraene, Pierre Francois, Francois Clad, and Pablo Camarillo Garvia. Topology Independent Fast Reroute using Segment Routing. Technical report, Internet Engineering Task Force, 2019.

[20] Gary Scott Malkin. RIP Version 2. STD 56, RFC Editor, 1998.

[21] John Moy. OSPF Version 2. STD 54, RFC Editor, 1998.

[22] Srihari Nelakuditi, Sanghwan Lee, Yinzhe Yu, Zhi Li Zhang, and Chen Nee Chuah. Fast local rerouting for handling transient link failures. *IEEE/ACM Transactions on Networking*, 15(2):359–372, 2007.

[23] D Oran. OSI IS-IS Intra-domain Routing Protocol. RFC 1142, RFC Editor, 1990.

[24] P Pan, G Swallow, and A Atlas. Fast Reroute Extensions to RSVP-TE for LSP Tunnels. RFC 4090, Internet Engineering Task {F}orce, 2005.

[25] Y Rekhter, T Li, S Hares, S Bryant, S Previdi, M Shand, G Enyedi, A Csaszar, A Atlas, C Bowers, A Gopalan, A Zinin, P Pan, G Swallow, A Atlas, M Shand, S Bryant, D Oran, John Moy, S Bryant, C Filsfils, S Previdi, M Shand, N So, and Gary Scott Malkin. A Border Gateway Protocol 4 (BGP-4). RFC 7811, RFC Editor, 1998.

[26] Retana Alvaro, Daniel Walton, Danny R McPherson, and Vijay Gill. Border Gateway Protocol (BGP) Persistent Route Oscillation Condition. Technical report, RFC Editor, 2002.

[27] John Scudder and Rohit Dube. Route Reflection Considered Harmful. Internet-Draft draft-dube-route-reflection-harmful-00, Internet Engineering Task Force, 1998.

[28] M Shand and S Bryant. IP Fast Reroute Framework. RFC 5714, RFC Editor, 2010.

[29] W Simpson. IP in IP Tunneling. Technical Report 1853, 1995.

[30] Phillip Smith. Weekly Routing Table Report. http://thyme.rand.apnic.net/current/data-summary.

[31] João Luís Sobrinho. An algebraic theory of dynamic network routing. *IEEE/ACM Transactions on Networking*, 13(5):1160–1173, 2005.

[32] João Luís Sobrinho, Laurent Vanbever, Franck Le, and Jennifer Rexford. Distributed Route Aggregation on the Global Network. pages 161–172, 2014.

[33] P Traina, D McPherson, and J Scudder. Autonomous System Confederations for BGP. RFC 5065, RFC Editor, 2007.

[34] Steve Uhlig and Sébastien Tandel. Quantifying the BGP routes diversity inside a tier-1 network. *Lecture Notes*

*in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3976 LNCS:1002–1013, 2006.

[35] Virginie Van Den Schrieck, Pierre Francois, and Olivier Bonaventure. BGP add-paths: The scaling/performance tradeoffs. *IEEE Journal on Selected Areas in Communications*, 28(8):1299–1307, 2010.

[36] Stefano Vissicchio, Luca Cittadini, Laurent Vanbever, and Olivier Bonaventure. iBGP deceptions: More sessions, fewer routes. *Proceedings - IEEE INFOCOM*, pages 2122–2130, 2012.