Réduire le manque de visibilité du système de routage publique dans l'Internet

Thomas Alfroy

THOMAS ALFROY

Reducing the Visibility Gaps of the Public Internet Routing System

---

Ph.D. Thesis

Université de Strasbourg

Université de Strasbourg

École doctorale Mathématiques, sciences de l'information et de l'ingénieur | ED 269 Université de Strasbourg

ECOLE DOCTORALE MATHEMATIQUES, SCIENCE DE L'INFORMATION ET DE L'INGENIEUR

Laboratoire ICube — UMR7357

THESE présentée par:

Thomas ALFROY

soutenue le: **11 septembre 2024**

pour obtenir le grade de: **Docteur de l'université de Strasbourg**

Discipline/Spécialité: **Informatique**

Reducing the Visibility Gaps of the Public Internet Routing System

Advisors

Cristel Pelsser (Director)	Professor, University of Strasbourg/UCLouvain

Invitee

Thomas Holterbach (Advisor)	Post-doc, University of Strasbourg

Co-Director

Pascal Mérindol	Associated Professor, University of Strasbourg

Thesis reviewers

Laurent Vanbever	Professor, ETH Zurich
Oliver Hohlfeld	Professor, University of Kassel

Jury president

Cedric Bastoul	Professor, University of Strasbourg/Qualcomm

Jury members

Cecilia Testart	Assistant professor, Georgia Tech
Dario Rossi	Professor, Telecom Paris/Huawei

Thomas ALFROY

Université de Strasbourg

Reducing the Visibility Gaps of the Public Internet Routing System

École doctorale Mathématiques, sciences de l'information et de l'ingénieur | ED 269 Université de Strasbourg

Résumé

Les plateformes de collecte BGP actuelles font face à un problème fondamental de gestion des données qui menace leur fonctionnement à long terme. Nous avons analysé, implémenté et évalué un nouveau paradigme de collecte de données, optimisé pour BGP. Notre système permet d'améliorer la collecte des données grâce à deux composantes : analyser la redondance dans les données BGP et utiliser cette dernière pour optimiser l'échantillonnage des données BGP collectées. Une définition appropriée de la redondance entre les routes BGP dépend de l'analyse effectuée. Nos contributions sont les suivantes : un sondage, des mesures et des simulations pour démontrer les limitations des systèmes de collecte actuels ; un système et des algorithmes génériques permettant d'évaluer et de supprimer la redondance dans les données BGP ; ainsi que des analyses quantitatives des bénéfices de notre approche en termes de précision et de visibilité pour diverses analyses de données BGP, telles que la détection des hijacks et la découverte de la topologie de l'Internet. Enfin, nous avons implémenté un prototype de collecte de données BGP qui automatise l'ajout de nouvelles sources de données. Le prototype est disponible à l'adresse https://bgproutes.io.

Résumé en anglais

BGP data collection platforms as currently architected face fundamental challenges that threaten their long-term sustainability. We analyze, prototype, and evaluate a new optimization paradigm for BGP collection. Our system scales data collection with two components: analyzing redundancy between BGP updates and using it to optimize sampling of the incoming streams of BGP data. An appropriate definition of redundancy across updates depends on the analysis objective. Our contributions include: a survey, measurements, and simulations to demonstrate the limitations of current systems; a general framework and algorithms to assess and remove redundancy in BGP observations; and quantitative analysis of the benefit of our approach in terms of accuracy and coverage for several canonical BGP routing analyses such as hijack detection and topology mapping. Finally, we implement and deploy a new BGP peering collection system that automates peering expansion using our redundancy analytics, which provides a path forward for more thorough evaluation of this approach. The prototype is available at https://bgproutes.io.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisors, Cristel and Thomas (affectionately referred to as TH by KC), for their unwavering support throughout this incredible journey. Words cannot fully capture how grateful I am for your guidance over the past three years. You were always there when I needed guidance, offering wisdom and insight whenever I faced challenges. A special thanks to Cristel, who believed in my potential from the very beginning and gave me the opportunity to embark on this Ph.D. journey. To Thomas, I will always remember those 3 a.m. calls before deadlines—moments that exemplify your extraordinary dedication. Both of you ensured that I was always in the best possible environment to succeed, teaching me countless things, both technical and non-technical, that I could never fully list here. This journey would not have been nearly as fulfilling without your mentorship. I have learned so much from both of you, and I sincerely hope that one day I can guide students with the same level of care and commitment that you have shown me.

I would also like to extend my gratitude thanks to my co-authors, Amreesh, Alberto, Thomas (another one!), and KC. Your contributions were invaluable, and this body of work would not have been possible without your support and collaboration. A special thank you goes to KC and the entire CAIDA group for warmly welcoming me during my four-month internship. Your expertise in Internet measurements was invaluable, and your feedback was always insightful. I am also deeply grateful for the opportunity to travel to the U.S. for the first time, where I made unforgettable memories.

I would like to express my sincere gratitude to the members of the network team at ICube, particularly Pascal and Jean-Romain, who mentored me during my master's thesis. Both of you ignited my passion for research, and I would never have embarked on this Ph.D. journey without your enthusiasm and encouragement. I am deeply thankful for all that you have done for me.

I also want to thank my friends, whose constant support over these three years has meant so much. To my family, who have been incredibly supportive (for much longer than three years!), I am endlessly grateful.

Lastly, a special thank you to Laura for her unwavering support. This three-year journey would have been far more difficult without your patience and kindness. Thank you so much!

Abstract

The expansion of the Internet enabled the emergence of various services. Nowadays, any user connected to the Internet can access video content posted from another country or communicate with other users located at the other end of the world almost instantly. All these services are accessible from everywhere on the Internet thanks to a protocol, BGP, which ensures the connectivity between all networks operating on the Internet. While BGP operates smoothly most of the time, it can be disrupted by various anomalies, ranging from simple outages to large-scale attacks. To ensure that the Internet operates smoothly, it is necessary to monitor BGP.

Multiple BGP data collection platforms, public or commercial, enable analyzing BGP data by collecting them from various networks operating on the Internet. However, current platforms face multiple limitations, often conflicting. First, these platforms collect data from a limited number of networks, reducing the visibility of the Internet routing ecosystem. While a simple solution could be to increase the coverage of these platforms, this approach combines with another limitation of these platforms, the volume of collected data. Specifically, even with the current limited coverage, the collected volume of data is prohibitive for users, which limits its practicability. Therefore, there is an urgent need to reevaluate the way we are collecting BGP data.

To this end, we propose GILL, a BGP data collection system that can collect data from an order of magnitude more Internet networks. This system leverages a key observation we made about BGP data: they often exhibit a high level of redundancy. GILL then uses this redundancy to collect only the most useful data from a larger number of networks. In this thesis, we detail GILL's design, as well as the benefits brought by such a technique to the scientific community, both in the short term and the long term.

Résumé

L'expansion de l'Internet a permis à de multiples services d'émerger. Aujourd'hui n'importe quel utilisateur peut accéder à du contenu vidéo posté depuis un autre pays ou échanger des messages avec d'autres utilisateurs à l'autre bout du globe en un clin d'oeil. Tous ces différents services sont rendu accessible partout dans l'Internet grâce à un protocole, BGP, qui assure la connectivité entre les différents acteurs de l'Internet. Bien que BGP opère correctement la majeure partie du temps, son fonctionnement peut être perturbé par diverse anomalies allant de la simple panne à l'attaque à grande échelle. Afin de s'assurer du bon fonctionnement de l'Internet, il est nécessaire de surveiller BGP.

Diverses plateformes de collecte de données BGP, publiques ou commerciales, permettent une analyses des données BGP en les collectant depuis différents réseaux opérant dans l'Internet. Cependant, les plateformes actuelles font face à plusieurs limitations, souvent conflictuelles. Premièrement, ces plateformes collectent des données depuis un nombre limité de réseaux, limitant la visibilité de l'écosystème de l'Internet. Bien qu'une solution simple serait d'augmenter la couverture des plateformes, cette approche entre en conflit avec la seconde limitation de ces plateformes, le volume de données collecté. En effet, même avec la couverture limitée qu'on les plateformes de collecte à l'heure actuelle, le volume de données collecté limite leur bonne utilisation par les utilisateurs. Ainsi, un besoin urgent de réévaluer notre manière de collecter les données BGP se fait ressentir.

Dans cette optique, nous proposons GILL, un système de collecte de données BGP qui permet de collecter des données depuis un ordre de magnitude supplémentaire de réseaux de l'Internet. Ce système se base sur une observation clé faite sur les données BGP: elle contiennent très souvent un niveau élevé de redondance. GILL exploite donc cette redondance afin de ne collecter que les données les plus pertinente depuis un nombre plus élevé de réseaux. Au cours de cette thèse nous détaillerons le design de GILL, ainsi que les bénéfices qu'une telle stratégie de collecte des données peut apporter à la communauté scientifique tant sur le court terme que sur le long terme.

Introduction

The Internet was originally designed as a network to facilitate communication between researchers from American universities. In the late 20th century, the Internet started gaining popularity among users worldwide, especially with the deployment of HTML in 1991. Today, many services run on top of the Internet, including critical services such as national defense or health information.

With nowadays billions of users accessing the Internet daily, most are aware of the risks they are exposed to by using the Internet: DDoS attacks, phishing, or various other scams. Therefore, typical end users have been incentivized to implement simple safety measures, such as using strong passwords or discarding suspicious emails. When I meet people and tell them my research works aim at improving the security in the Internet, they often ask if I can hack Facebook accounts (in which case I answer no!). Even those who are unfamiliar with computer sciences use the Internet daily for various purposes, yet few truly understand how the Internet practically works. People use the Internet to interact with various services or other individuals without knowing how these connections are made. Most are unaware of the concept of routing, which consists of finding the best route between two end points of a network. They thus ignore the existence of the Border Gateway Protocol, cornerstone of the Internet [1].

With nowadays more than 75 000 networks operating in the Internet, routing is ensured in an automatic fashion by a scalable protocol called the Border Gateway Protocol (BGP). Despite being the glue between all the networks of the Internet, BGP is far from being bulletproof. BGP was not initially designed with security in mind, as its widespread adoption was unexpected. Consequently, anomalies disturbing the smooth operation of the Internet regularly occur, reducing the quality of the user's experience. In 2021, we can cite a large-scale disruption of Instagram, Facebook, and WhatsApp services resulting from a BGP misconfiguration [2]. Such events often make the headlines and highlight the urgent need to improve and monitor the routing security on the Internet.

Monitoring the Internet has become a significant research field, leading to numerous efforts made to understand and improve the Internet ecosystem. The importance of monitoring the Internet routing ecosystem lead to the emergence of companies dedicated to this task as their primary business. Monitoring the Internet involves

analyzing *BGP routes*, which are collected by BGP data collection platforms such as RIPE RIS [3] or RouteViews [4]. These platforms collect data from different networks worldwide, known as Vantage Points (VPs), and have been publishing this data since 2000. Despite their extensive use by both researchers and network operators, these platforms exhibit multiple and sometimes conflicting limitations.

What are the limitations of the current BGP data collection platforms? BGP, by design, inherently hides some information about the underlying topology, resulting in a VP only having a partial view of the Internet routing ecosystem. Nowadays, public BGP data collection platforms collect data from roughly 1% of the networks operating on the Internet. However, this coverage is too low and leads to many routing information being off the radar [5].

Why a low coverage is a problem? Many measurement analysis and monitoring tools rely on the collected BGP data. Since each VP only provides a partial view of the Internet routing ecosystem and the platforms cover only 1% of the networks operating in the Internet hosting a VP, the collected data is incomplete. This can result in many routing anomalies remaining off the radar, or the results of measurement analysis being inaccurate. Attackers can thus cause data leaks, loss of connectivity, or cryptocurrency hijacks among other undesired effects without being detected by monitoring systems. In this thesis, we take a first step toward evaluating the gap in routing visibility induced by low coverage of BGP data collection platforms using simulations. We demonstrate that the current coverage of the BGP data collection platforms leads to missing many BGP routing information.

Why has the coverage remained flat over the years? A straightforward solution might be to increase the number of VPs. Although this would effectively improve the coverage of the Internet ecosystem, it also poses technical challenges induced by the high volume of data collected by these platforms. As a result of this limitation, the coverage of the BGP data collection platforms has remained flat over the years, limiting our visibility of the Internet routing ecosystem and the effectiveness of the analysis relying on this data.

What are the current solutions? Several solutions have been proposed to enhance our knowledge of the Internet topology. Researchers have attempted to determine in which network of the Internet deploying a VP would maximize the tradeoff between additional unique information and the volume of data collected [6, 7, 8]. Other researchers have used active measurements as an additional source of data, alongside the BGP data. However, none of these strategies are sustainable. In this thesis, we demonstrate that every network on the Internet contributes a (sometimes small) piece of unique data. Therefore gathering all possible BGP information requires collecting

data from all networks operating on the Internet. A drastic increase in the number of networks collecting BGP routing data comes alongside a significant increase in the volume of collected data. Similarly, using active measurement to supplement routing data is impractical, as forwarding paths have been proven not to necessarily reflect routing paths [9]. In addition most of the possible paths on the Internet are inactive, since they are only used after a failure.

Therefore, we present the main contribution of this thesis, GILL, a BGP data collection system that can cope with the data management challenges induced by a drastic increase in the VP coverage. GILL leverages a key observation we made about the BGP data: it often exhibits a high level of redundancy. GILL implements two carefully selected definitions of redundancy, as we demonstrate that naive or specific definitions of redundancy may lead to an ineffective sampling for many possible objectives. To define the redundancy in BGP data, we propose two algorithms: BUS (BGP Update Selector) and MVP (Most Valuable Vantage Points). BUS assesses the redundancies at the update granularity, while MVP assesses the redundancy at the VP granularity. GILL then uses the computed redundancies to build filters that retain only the most useful bits of data.

Outline. We divide this thesis in eight different chapters.

In §1, we introduce the process of routing on the Internet and present the different characteristics of BGP. We then highlight the importance of monitoring and analyzing BGP by presenting numerous analysis that rely on BGP data collected by public collection platforms. We finish this first chapter by detailing how BGP data is collected in the Internet.

In §2, we present the current BGP data collection systems, their limitations, and the underlying challenges of scaling them. We take a first step toward estimating the visibility gap of current data collection platforms by using simulations in a controlled environment. Additionally, we present the challenges users face when dealing with BGP data by conducting a survey among authors of scientific papers published in computer network conferences.

In §3, we propose a comprehensive analysis of the redundancy in the BGP data. We start by illustrating, using a simple gadget, how BGP data can be redundant. We then introduce three gradually stricter definitions of redundancy that enable us to highlight the high level of redundancy present in BGP data. While these three definitions enable us to quantify the redundancy in BGP data, none of them are used in any of our system's designs.

In §4, we explore the effectiveness of different redundancy definitions. We demonstrate that naive definitions often fail to provide efficient sampling for various objectives. Therefore, we present two main contributions of this thesis: two algorithms that evaluate the redundancy at two different granularities, enabling efficient sampling regardless of the user's objective. BUS leverages a new metric, the *reconstitution power*, to sample BGP data at the granularity of the BGP message. MVP evaluates and compares the partial view of each VP by computing the impact of BGP routing events on these partial views using topological features. This algorithm creates a set of the least redundant VPs.

In §5, we describe the main contribution of this thesis, GILL, a system that improves the BGP data collection by focusing only on the most useful bits of data. GILL relies on an overshoot-and-discard collection strategy, meaning it collects data from as many VPs as possible but discards the redundant portion of the data before storing it. GILL uses the redundancy definitions provided in §4 to build filters that discriminate redundant data from non-redundant data. We detail and motivate all the design choices of GILL. Finally, we present the implementation of GILL, which is currently up and running.

In §6, we evaluate GILL's sampling on various use cases. We start by benchmarking GILL's sampling against naive and specific baselines, illustrating GILL's capability to provide efficient sampling and avoid overfitting toward any specific objective. We then present the long-term impact of GILL using controlled simulations. Next, we describe the short-term impact of GILL's sampling, showing how it can improve the accuracy of common measurement studies. Finally, we present the impact of GILL's sampling on DFOH a system that samples BGP data using GILL's definitions and detects forged-origin hijacks in the wild.

In §7, we present potential future works to improve BGP data storage and processing.

Finally, in §8, we conclude this thesis and provide potential guidelines to incentivize network operators to establish a peering session with GILL.

Publications

Most of the work presented in this thesis has been presented in conference proceedings or workshops. Below is presented the list of accepted publications.

Conference publications

The Next Generation of BGP Data Collection Platforms.
Thomas Alfroy, Thomas Holterbach, Thomas Krenc, KC Claffy and Cristel Pelsser, In *Proc. of ACM SIGCOMM 2024*, Sydney, Australia.
🏆 **SIGCOMM best paper award**

A System to Detect Forged-Origin Hijacks.
Thomas Holterbach, Thomas Alfroy, Amreesh Phokeer, Alberto Dainotti and Cristel Pelsser, In *Proc. of USENIX NSDI 2024*, Santa Clara, USA.

Workshop publications

Internet Science Moonshot: Expanding BGP Data Horizons.
Thomas Alfroy, Thomas Holterbach, Thomas Krenc, KC Claffy and Cristel Pelsser, In *Proc. of ACM HotNets 2023*, Boston, USA.

Posters and Technical reports

Measuring Internet Routing from the Most Valuable Points.
Thomas Alfroy, Thomas Holterbach, Thomas Krenc, KC Claffy and Cristel Pelsser, *Arxiv preprint, 2024*, https://arxiv.org/abs/2405.13172.

MVP: Measuring Internet routing from the most valuable points.
Thomas Alfroy, Thomas Holterbach and Cristel Pelsser, In *Proc. of ACM IMC 2022*, Nice, France.

Developped tools

Implementation of GILL, the next generation of BGP data collection platforms. This implementation comprises a custom BGP daemon, an orchestrator in charge of computing the filters and a web interface, https://bgproutes.io/

Implementation of DFOH, a system to detect forged-origin hijacks. This implementation comprises the inference pipeline and a we interface, https://dfoh.uclouvain.be/.

Contents

1

Background

In 1837 the first commercial system of wire communication using electrical signals, the *Cooke and Wheatstone System* [10] was designed. The system operates simply: it uses a domestic telegraph instrument that the owner connects to the city's switchboard office. When trying to contact a given destination, the instrument first reaches the local switchboard office. The switchboard operator then manually forwards the traffic to the local switchboard office of the destination. Finally, the switchboard operator manually connects the intended destination. With this system, it was possible to send around 30 words per minute.

Although nowadays communication networks exhibit significantly higher performances, the underlying principle remains the same: end nodes interconnected by network relays providing transit. The traffic transits node by node until it reaches the final destination. The process of finding a path from one node to another is called *routing*. At the early stage of communication networks, routing between nodes was performed manually by a switchboard operator. With today tens of billions of connected devices [11], manual routing is no longer feasible. Consequently, communication networks rely on *routing protocols* to automatically find routes between nodes.

This chapter explores how connectivity is enabled at the Internet scale and the challenges related to Internet routing. We start in §1.1 by describing how routing protocols

practically work. We explain in §1.2 how routing is operated at the Internet scale. Next, we examine in §1.3 the different challenges related to Internet routing. Finally, we present in §1.4 how the Internet routing ecosystem is monitored.

1.1 Routing in a network

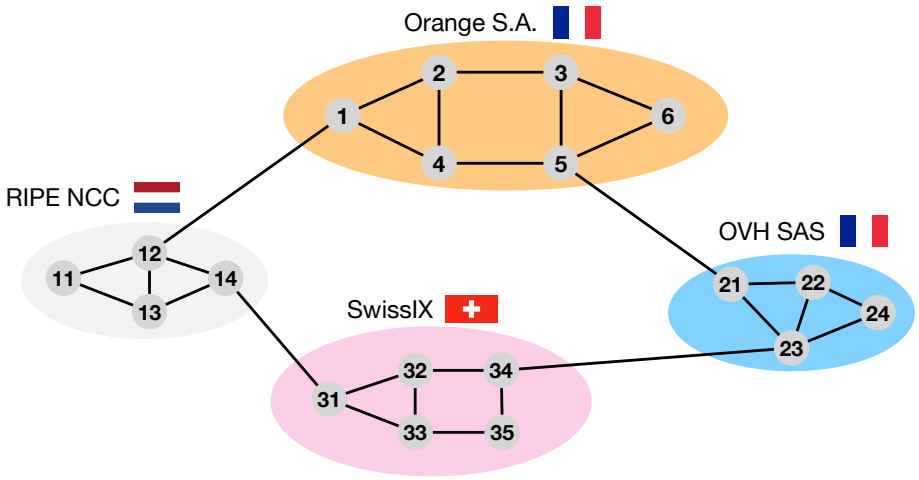


Figure 1.1: Example of network topology, represented as a graph with nodes and links. The network is divided into sub-networks, each one handled by a single administrative entity.

A communication network is a set of nodes, called routers, connected via physical wired links. Routing is a process that enables finding a path between each pair of nodes. For instance, in Fig. 1.1, a possible path from 1 to 5 is 1—2—3—5. To prevent any single point of failure, network operators often add redundancy to their topology. In Orange network [1] (Fig. 1.1) there is no possible failure on a link that will make any node unreachable.

Topology redundancy not only provides resilience against failures but also offers multiple choices of paths between each pair of nodes. Routing enables finding not only one path between each pair of nodes but the *best path*. The best path definition may depend on various criteria, such as the link capacity, latency, or routing policies. A naive criterion may be the number of traversed links, in which case the best path from 1 to 5 is 1—4—5.

Routing can be operated either statically or dynamically. Static routing involves manually installing rules in the routers that enable directing the traffic on the best path.

[1] The network topologies are not representative of the reality.

Network operators from Orange (Fig. 1.1) may install the following rule in router 1 : "*Forward the traffic destined to* 5 *to* 4". While static routing can be implemented on small-scale networks, it becomes quickly challenging for large-scale networks. Beyond this operational challenge, static routing may fail because of frequent topological changes occurring within the network. If link 4—5 fails, 1 loses its connectivity to 5. Dynamic routing enables an automatic selection of the best paths, based on the underlying topology. It is responsive to topological changes, i.e., the paths are automatically updated according to the topological dynamics. If the link 4—5 fails, dynamic routing will automatically install a rule in router 1 that forwards the traffic destined to 5 to 2 — the new best path after the failure. Dynamic routing is efficient for large-scale networks, prevents negative impacts of routing anomalies such as forwarding loops or dead ends, and enables exploiting the network capacities.

Dynamic routing is implemented by routing protocols. A routing protocol is a standardization of messages exchanged by routers, allowing a dynamic best-path selection. The best paths are stored in a *routing table*, a structure that maps every destination to the corresponding shortest path. There exist three main categories of routing protocols: *link-state*, *distance-vector*, and *path-vector*. The common point between all of these protocols is their reliance on shortest-path algorithms, such as the Dijkstra [12] and the Bellman-Ford [13] algorithms.

Distance-vector protocols implement a distributed version of the Bellman-Ford algorithm. Each node in the network advertises its distance to all other nodes to its neighbors. Upon reception of a message, a router recomputes its best path and updates its routing table, if a shortest distance is received. The main drawback of distance-vector protocols is the convergence time, particularly in large networks, since each router must send and transmit the distances of all other routers.

Link-state protocols enable routers to exchange information about their connectivity until every router in the network is able to build a map of the topology. This map takes the form of a directed graph where the nodes are the routers connected by physical links. Each link is weighted according to any performance metric such as the link capacity or the latency. Each router then applies the Dijkstra algorithm on the topology, using itself as a source to compute the best path to every other router in the network. The most well-known link-state protocols are OSPF [14] and IS-IS [15]. While the complexity of the Dijkstra algorithm remains low $O(|links| + |nodes|\, log\, |nodes|)$, these protocols still do not scale for networks with tens of thousands of nodes experiencing frequent topological changes. They are only used to enable connectivity within a network, i.e., for *intra-domain* routing.

Path-vector protocols rely on the principles of the Bellman-Ford algorithm, where each router advertises its best path to all the destinations to its neighbors. The most famous path-vector protocol is the Border Gateway Protocol (BGP) [1] and is the cornerstone of the Internet.

1.2 Routing a Internet Scale

The Internet is a network of networks, where connectivity is enabled by inter-domain routing protocol. The Internet topology is commonly represented as a graph where nodes are Autonomous Systems (ASes), i.e., individual networks operated by a single administrative entity and links are connections (physical or logical) between these ASes. For instance, in Fig. 1.1, the AS-level topology graph is composed of four nodes (Orange, RIPE NCC, SwissIX and OVH) and four links. The connectivity within each of these nodes (AS) is operated by intra-domain routing protocol, while connectivity with other nodes requires inter-domain protocol. Each AS in the Internet is identified by a unique Autonomous System Number (ASN) and owns one or more *prefixes*, i.e., sets of contiguous IP addresses. Today, the Internet is composed of more than 75 000 ASes which globally announce more than one million prefixes [16]. Because of the size of the Internet and the frequent topological changes, distance-vector, and link-state routing protocols exhibit scalability problems.

The de facto protocol used in the Internet is the Border Gateway Protocol (BGP) [1], categorized as a path-vector protocol. In BGP, an AS announces each prefix it owns using a BGP message sent to its neighboring ASes. This announcement is subsequently propagated AS by AS until every AS on the Internet has a route to reach the announced prefix. Unlike intra-domain routing, inter-domain routing is used to enable connectivity between networks that are not under the same administrative authority. Consequently, the best path selection cannot rely on performance metrics such as latency. The Internet is a business that exhibits a hierarchical pattern with a few ASes at the top (called *Tier1*), where some ASes are paid by customer ASes to provide transit. BGP is thus not only destination-based (prefixes) but also policy-based. The best path selection predominantly relies on the routing policies and the economical relationships between ASes.

Routing policies on the Internet. The Internet is a business where the best path selection depends on the routing policies. These routing policies are defined by an AS according to the economic relationships it maintains with neighboring ASes. There are three main types of AS relationships [17, 18], defined as follows:

- **Customer-to-Provider** (c2p): a customer AS pays an upstream AS to forward its traffic on the Internet.

- **Peer-to-Peer** (p2p): two ASes mutually agree to exchange traffic for free.
- **Sibling-to-Sibling** (s2s): two ASes that are owned by the same company agree to exchange traffic for free.

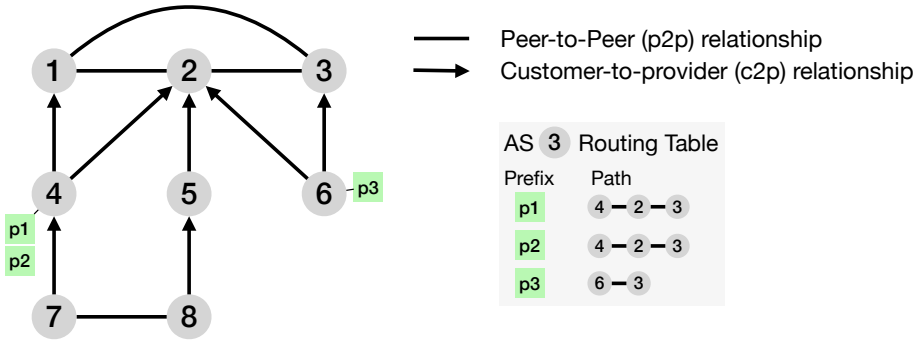


Figure 1.2: Example of an mini-Internet topology. The arrows represent c2p relationships, while lines represent p2p relationships.

Fig. 1.2 represents an Internet topology of eight ASes. P2p relationships are represented by a line, whereas c2p relationships are represented by an arrow. *AS4* announces two prefixes, p1 and p2 and *AS6* announces p3. The first criterion of best path selection is thus not any performance metric, but rather the relationship with the AS you receive the route from. Also, a route received from a neighbor is not blindly propagated to other neighboring ASes. In Fig. 1.2, if *AS7* receives a route to p1 from *AS4*, it will not advertise this route to its neighbors with which it maintains a p2p relationship. In fact, *AS7* will pay to provide transit to an AS that is not one of its customers. Gao et al. proposed in 2001 the *Gao-Rexford model* [19], a set of export policies that enable consistent routing in the Internet by having only *Valley-free* paths:

- Routes learnt from providers or peers are only advertised to customers.
- Routes learnt from customers are preferred over routes learnt from peers. Routes learnt from peers are preferred over routes learnt from providers.

To implement these rules, AS uses the *local-pref*, a numeric value that can be attached to a route. Operators configure this value such that a route received from a customer is attached a higher local-pref than a route received from a peer. If a BGP router receives two different routes to the same prefix, it will select the route with the highest local-pref. While these rules allow consistent export policies, they are not sufficient to enable the best path selection upon the reception of a new route. *AS6* will receive two distinct routes to prefix p1; 4—2—6 and 4—1—3—6 which are both valley-free and

received from a provider. Therefore, BGP needs other mechanisms to ensure the best route selection for every prefix.

BGP route attributes. BGP is a protocol that runs on top of Transmission Control Protocol (TCP). Upon the establishment of the TCP session between the routers of the two neighboring ASes, they both send an *Open* message that enables exchanging information about the AS running BGP, such as the ASN or the router capabilities. Once the BGP session is established, the two routers start exchanging BGP routes using *BGP updates* messages. For each prefix in the routing table (called Routing Information Base or *RIB* in BGP), a BGP router sends its best route, using a BGP update. Upon receiving a new route, a BGP router compares it to the route stored in the RIB and replaces the older one if the new one is considered as better. Only the new best route will be then announced to the neighboring ASes (with respect to the export policies). To compare two routes, a BGP router relies on *BGP attributes* contained within a BGP update (in case the local-prefs are identical). One of the main BGP attributes is the *AS path*, the list of ASes that the route has traversed, identified by their ASN. The best route to a given prefix is the one with the shortest AS path. The best route from *AS6* to p1 is then 4—2—6. There exist other attributes such as the *BGP communities*, a set of couples of numeric values that give information on how an AS receiving this route is supposed to process it or encode various information about the route. This BGP attribute has been widely studied and enhances our understanding of the Internet routing ecosystem [20, 21, 22, 23]. While there exist other BGP attributes, most of the time they are omitted from the measurement analysis. Therefore, we will focus on the AS path and the BGP communities.

Although the Internet operates (most of the time) smoothly, its routing ecosystem remains pretty opaque and opens the opportunity for many measurement analysis. These analysis are essential for enhancing our understanding of the routing ecosystem and improving its reliability.

1.3 BGP challenges

BGP has been extensively studied over the last three decades. Many efforts have been made to understand the Internet routing ecosystem and make it more reliable. Researchers have focused on various aspects including analyzing the *Internet topology*, examining the *BGP convergence*, measuring the impact of *large-scale outages and misconfigurations* on BGP routing, detecting and mitigating *BGP routing attacks*, or enhancing our understanding of the *BGP communities*. Although the scope of BGP-related studies is broad and cannot be exhaustively presented, we review some key studies in this area.

1.3.1 Mapping the Internet Topology

A BGP router does not have the full visibility over the Internet topology. There are two main reasons behind this limitation: *(i)* because of the routing policies, not all the routes are propagated upon reception, *(ii)* BGP only sends the best route to each prefix, limiting the visibility of backup links.

Measuring the gap in the visibility of AS topology. Over the decades, numerous research works have been made to discover what we are missing from the AS topology. For Instance, Roughan et al. used Expectation Maximization algorithms to estimate the proportion of missing links [24]. Their measurements indicate that more than 60% of the p2p links are unobservable. Chang et al. used multiple less-studied data sources to evaluate the proportion of missing links [25]. They used Internet Routing Registries as well as traceroute data and found similar results than [24]. Muhlbauer et al. [26] improved the model by removing the assumption that an AS in the topology is an atomic structure. Numerous other efforts have been made to estimate the gap in visibility in the Internet topology, e.g., [27, 28, 29, 30]. Despite using different strategies to achieve the same goal, all these studies rely on external sources of data such as IXP data or traceroute measurements. Donnet et al. published a comprehensive survey on the different techniques that have been developed [31].

Inferring the AS relationships. The latest survey also considers another fundamental measurement challenge: inferring the AS relationships. Gao' pioneering work [32] addressed this challenge by using a heuristic algorithm based on export policies and AS degrees extracted from AS paths to infer the relationships implemented by each AS link . Gao validated her results using internal data from a large Internet Service Provider (ISP). Following Gao's work, Subramanian et al. proposed a formalization of the AS relationship inference problem [33]. Dimitropoulos et al. improved AS relationship inferences by assuming that not all the paths observed on the Internet follow a valley-free pattern [34]. They also improved the validation process by surveying 78 ISPs to validate their accuracy, ending up with 94.5% of the links correctly inferred. In addition to improving the AS relationship inference, Luckie et al. proposed a new strategy for obtaining a more complete ground truth for validation [35]. Building on Quoitin et al.'s finding that BGP communities can indicate AS relationships [36], Luckie et al. used data from IRR to get information about BGP communities and parsed the import- export policies to determine AS relationships. They were able to validate 34.6% of their inference, by far the largest number of validated AS relationships. AS relationship inference has been extensively explored with many efforts aimed at solving this challenge (e.g., [37, 38, 39, 40, 41]). Having a better understanding of the AS relationships is an important step toward a more transparent Internet routing ecosystem and the results of these algorithms have been heavily used in other analysis.

1.3.2 Analyzing BGP convergence

Because BGP is a path-vector protocol, it requires sending route updates to the neighboring ASes, which then propagate to the entire Internet. This distributed version of the Bellman-Ford algorithm has a drawback: the convergence time.

Measuring the convergence delays. Over the last two decades, numerous efforts have been dedicated to understanding, measuring, and improving the inter-domain convergence. Labovitz et al. studied the convergence time after an inter-domain failure [42]. They found that, unlike intra-domain which converge in a few milliseconds, inter-domain convergence can take up to ten minutes. Griffin et al. employed statistical models to conduct experimental analysis on the BGP convergence time [43]. They uncovered convergence issues and showed, using simple examples, that in some cases BGP may not converge because of loops in BGP announcements [44]. There exist other works focusing on studying BGP convergence after a failure (e.g., [45, 46]).

Improving BGP convergence. Several works have also focused on improving the BGP convergence [47, 48]. For instance, Pei et al. infer the root cause of an inter-domain failure to accelerate the convergence by selecting only the paths that are not impacted by the failure [49]. Holterbach et al. proposed Swift, a system designed to improve the convergence after an inter-domain failure [50]. They detect and use bursts of route withdrawal to infer where the failure occurred in the topology and find alternative paths that circumvent the failure. Additionally, they use programmable dataplane to accelerate the process and make it more practical.

Studying the convergence of BGP is essential to improve the performance and the reliability of the Internet.

1.3.3 Impact of large scale events on BGP routing.

Despite not targeting Internet routing directly, some large-scale events had a significant impact on BGP.

Large-scale attacks on services. Large-scale events like worm attacks can significantly impact BGP routing. In 2003, the SQL Slammer worm exploited a bug in MS SQL servers [51]. Although this attack was not directly targeting routing infrastructures, a significant spike in the number of BGP messages exchanged was noticed [52, 53]. This increase was induced by a high number of AS links operating above the critical load thresholds, causing routers to restart and repeatedly resend their entire RIB. This is

not an isolated case of large-scale attack disrupting the BGP routing, we can also cite the Code-Red attack in 2001 [54, 55] or the NIMDA attack in 2001 [56, 57].

Non-routing events. Other events not targeting specifically routing infrastructures such as physical outages due to natural disasters can impact inter-domain routing. The Great East Japan Earthquake and Tsunami in 2011 caused inter-domain discrepancies [58] by breaking a significant number of physical links, resulting in BGP sessions being shut down. Carisimo et al. conducted a longitudinal analysis of the Venezuelan Crisis [59]. They studied the impact of the crisis on the BGP announcements, among other effects. Researchers also studied the impact of other headline-making events on the Internet [60, 61].

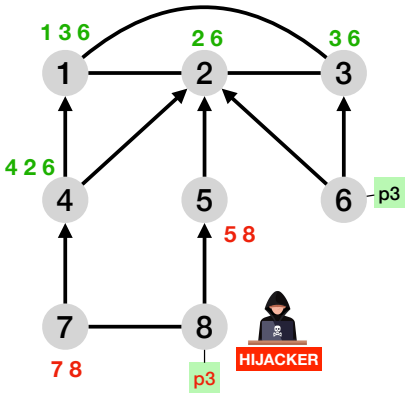
Large-scale misconfigurations. The Internet is a distributed system operating without central authority. Each AS can administrate its network, and thus its inter-domain routing policies without any global consensus. While inter-domain routing works generally smoothly, occasional mistakes happen when network operators configure their BGP routers. One noticeable event occurred in 2008 when Pakistan tried to block YouTube inside their country but inadvertently misconfigured their network, causing the traffic to many central destinations to be diverted and dropped, affecting to a large portion of the Internet connectivity [62].

Studying these events and their impact on inter-domain routing is essential to enhance our understanding of the routing ecosystem and prevent their recurrence in the future.

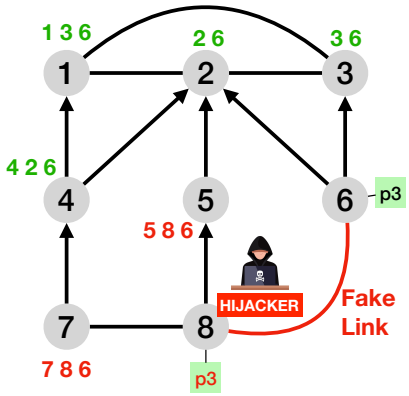
1.3.4 Decting and mitigating routing attacks

Despite being the cornerstone of the Internet, BGP was not designed with security in mind: Internet routing remains vulnerable to *BGP hijacks*. A BGP hijack occurs when an attacker creates a fake BGP announcement to divert a portion of the traffic from its legitimate destination.

Detecting Multiple Origin ASes hijacks. In Fig. 1.3a, *AS8* executes a BGP *Multiple Origin ASes* hijack, one of the most common attacks that occurs regularly in the Internet routing ecosystem [63]. This hijack consists of an AS announcing a prefix it does own legitimately. This hijack propagates to the neighboring ASes and redirects the traffic from ASes whose route to the attacker is preferred over the legitimate one to the attacker. This type of hijack has fueled an entire research area, with numerous efforts aimed at detecting and mitigating these attacks [64, 65, 66, 67, 68]. For instance ARTEMIS is a system deployable by an AS and capable of detecting these hijacks and proposing mitigation guidelines [69]. Similarly, by providing information about an



(a) Example of Multiple Origin AS hijack. *AS8* hijacks prefix p3.



(b) Example of a Forged-Origin hijack. *AS8* hijacks prefix p3 by prepending the legitimate origin, *AS6*.

Figure 1.3: Example of BGP vulnerability: the BGP prefix hijack.

AS's prefixes, BGPalerter can send notifications to the network operator upon detection of a MOAS hijack [70].

Measuring the impact of Multiple Origin ASes hijacks. Numerous measurement analysis have been conducted to assess Internet vulnerability to MOAS hijacks. Apostolaki et al. proved that attackers can intercept a fraction of the BitCoin traffic by using BGP hijacks [71]. They demonstrated the feasibility of such attacks and showed that by hijacking around 100 prefixes, an attacker can divert 50% of the BitCoin traffic. Testart et al. tracked *serial hijackers*, i.e., ASes that regularly appear to be at the origin of a BGP routing attack [72]. They built a classifier that identified 900 ASes from the IPv4 routing table with suspicious behaviors such as misconfigurations or benign hijacks. Cho et al. used machine learning to classify BGP hijacks into different categories [73]. All these works contribute to enhance the scientific community's understanding of the impact of BGP MOAS hijacks.

To enable a BGP router to drop a fake route induced by a MOAS hijack, a new feature was proposed and plugged into most of the BGP implementations: RPKI [74].

Preventing Multiple Origin ASes hijacks with RPKI. Resource Public Key Infrastructure (RPKI) is an extension of BGP that enables detecting BGP routes where the origin AS and the prefix do not match [74]. It operates as a cryptographically-based database where ASes can register the prefixes they own using their public key. A router that deploys RPKI validation uses this database to identify and discard illegitimate routes. Numerous analysis have been conducted to measure RPKI database coverage and route filtering deployment based on RPKI [75, 76, 77, 78]. Li et al. developed a

tool, RoVista, which measures in real-time the proportion of invalid routes filtered by ASes [79]. The state-of-the-art techniques usually involved a controlled prefix illegitimately announced followed by HTTP requests to determine which ASes ingested the illegitimate route. Li et al. used the IP-ID field of the IP header as well as IP spoofing to measure the connectivity toward RPKI invalid prefixes.

Detecting Forged-Origin hijacks. While a full deployment of RPKI may enable avoiding MOAS hijacks, BGP remains vulnerable to *Forged-Origin hijacks*. To execute such an attack, the hijacker announces a prefix it does not own but prepends the legitimate origin of the prefix, rendering the fake route undetectable by RPKI validation. In Fig. 1.3b, *AS8* hijacks p3 and prepends *AS6* to the AS path. This attack is actively used by hijackers and made the headlines several times [80, 81]. These hijacks often induce a new *fake link* between the attacker and its victim, as we can see in Fig. 1.3b. However, inferring the legitimacy of an AS link is challenging given our incomplete knowledge of the AS topology and the regular appearance of numerous new legitimate AS links. Holterbach et al. developed DFOH, a system designed to detect forged-origin hijacks [82]. DFOH extracts topological features from the AS topology to estimate the extent to which the new link breaks the hierarchical structure of the Internet. DFOH also uses features extracted from PeeringDB [83], a database populated by network operators about their connectivity. In peeringDB, operators can register their presence in *Internet Exchange Points* (IXP) which serve as facilities where ASes can connect to other participants without requiring a direct physical connection to them. Leveraging these features, DFOH trains a Machine Learning model and achieves a 95% detection accuracy.

1.3.5 Studying BGP communities

While not directly used in the best path selection process, BGP communities are valuable sources of information about the routing ecosystem. They are used to tag routes, influence the route selection process, or enforce some routing behaviors.

Uncovering the use of BGP communities. Several works have tried to uncover the role of BGP communities and give a better understanding of their use. Quoitin et al. demonstrated that communities can be used for traffic engineering purposes, i.e., to influence routing decisions [84]. For instance, a community may indicate the provider to prepend its AS, resulting in a less attractive route. They also demonstrated that a community can be used to tag a route, for instance indicating that the route has been received on a p2p link.

Classifying BGP communities. Several efforts have been made to classify the communities according to their use. Donnet et al. [23] proposed a classification of

BGP communities comprising three categories: *Inbound*, used upon reception of a new route; *Outbound*, used to influence route propagation; and *Blackhole*, used by an ISP to block packets. Krenc et al. [20] went one step further and designed a probabilistic algorithm capable of classifying BGP communities into two categories *(i) information* communities, used to tag a route; and *(ii) action* communities, used to influence the routing decisions. They leveraged a clustering strategy as well as a new metric the *on-path:off-path ratio* to achieve a classification accuracy of 96.5%.

All these studies enhance our understanding of the Internet routing ecosystem. Despite their wide range of focuses, all BGP-based studies use the same data source: the BGP routing data collected by *BGP Vantage Points*.

1.4 BGP Vantage Points in a nutshell

To conduct BGP-related studies, researchers need authentic BGP data that reflects the real Internet routing. Most of them are using the BGP data collected from *BGP Vantage Points* (VPs). A VP is a router within an AS that agrees to export its routes to a *BGP route collector*. BGP route collectors are deployed by BGP data collection platforms among which RIPE Routing Information Service (RIS) [3] and RouteViews (RV) [4] are the most widely used.

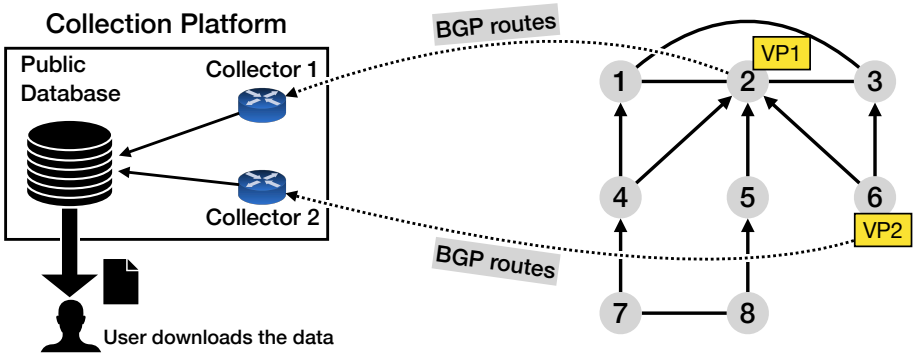


Figure 1.4: Example of BGP data collection system. VPs export their BGP route to BGP collectors.

Fig. 1.4 describes a BGP data collection system. In this example, two ASes host a VP, *AS2* and *AS6*, meaning that one router within each of these ASes maintains a peering session with a BGP collector. A BGP route collector is a router or a server hosted by a collection platform running a BGP daemon, such as FRR[1] for RV. Each BGP route collector can have one or more sessions with different VPs. The data collected from all

[1] FRRouting, routing daemon

the collectors is then gathered into a public database, which the users can leverage for measurement analysis or anomaly detection for instance.

Usually, BGP data collection platforms provide two types of information: the raw *BGP updates* or *RIB dumps*. Raw updates correspond to all the collected BGP routes, while RIB dumps correspond to periodic snapshots of the RIB from each VPs. For instance, RIS provides a RIB dump every 8 hours. Both types of data are stored using the *Multi-Threaded Routing Toolkit* (MRT) format [85] and can be parsed using tools such as BGPstream [86] or MRT# [87].

1.4.1 BGP data collection platforms

In §1.3, we showed that researchers actively use BGP routing data from RIS and RV to conduct measurement analysis. Aside from RIS and RV, there exist other public BGP data collection platforms.

Public BGP data collection platforms. The two most well-known public BGP data collection platforms are RIPE RIS and RV. RIS was started in 1999 and focused on collecting BGP data from European ISPs. Over time, RIS increased its coverage and now has approximately 1500 VPs across the world [88]. They also proposed solutions for *remote peering*, enabling VPs to establish a peering session with the route collector without being directly connected, or having a presence in the same IXP. Similarly, RV is a project started in 2001 by the University of Oregon. Originally, it collected data from US ISPs but akin to RIS, it has today around 1000 VPs worldwide [89]. These two platforms are the most popular for two main reasons: *(i)* they are the most well-maintained, and *(ii)* they are both supported by common BGP data collection software, such as BGPstream [86]. There exist other BGP route collection platforms. For instance, *Packet Clearing House* (PCH) is sometimes used by researchers (e.g., [90, 91, 92, 93, 94]), but has not gained much traction partially due to its lack of compatibility with BGPstream. BGPwatch [95] operates 15 VPs and reports the potential routing attacks on their website. However, the collected BGP updates are not accessible in an MRT database. Finally, Isolario [96] used to collect data from VPs worldwide, but is not maintained anymore.

Despite being public platforms, they also are extensively used to fuel commercial BGP anomaly detection tools [97, 98].

Commercial monitoring tools. Several monitoring tools rely on BGP data to monitor some prefixes of the Internet. All these commercial tools use private BGP VPs, but some of them also use BGP data from RIS and RV. Kentik Network Observatory Platform [97] and ThousandEyes BGP monitoring platform [99, 100], although not disclosing their number of VPs publicly, said that they are using both public and private BGP

feeds. Radar by Qrator [101] uses 850 VPs and provides monitoring to ISPs. The system is able to detect routing incidents such as route leaks, hijacks or propagation of RPKI invalid prefixes. PacketVis [102] is a monitoring platform peering with ≈ 2000 peers, primarily sourced from RIS and RV. They enable the detection of hijacks, detection of peer change, and prefix monitoring for their clients. Finally, bgp.tools [103] is a system that collects data from ≈ 1000 private VPs. They provide monitoring for all the prefixes of the client AS and are able to detect MOAS hijacks, failures, and routing policy violations.

1.4.2 BGP Vantage Point characteristics

BGP VPs are a valuable source of information to measure the Internet routing ecosystem. While a VP may provide important information, it suffers from the native limitation of BGP: it only has a partial view over BGP events.

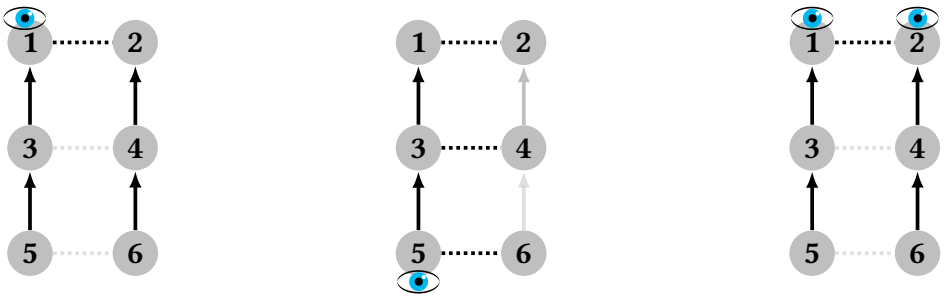


Figure 1.5: Combining local views can help map AS topology. Gray links are not visible from routes collected by VPs (👁).

BGP VPs have a partial view of the Internet. Since a VP is a BGP router that exports only its best route for each prefix to the route collector, each VP only has a local view of the routing ecosystem. We illustrate this using Fig. 1.5 which represents a small AS topology with six ASes and where the routing policies are configured to follow the Gao-Rexford model [19]. The dotted lines represent a p2p relationship, while the arrows represent a c2p relationship. Let's suppose a simple model where every AS announces one prefix and operates one BGP router. In Fig. 1.5a, the VP deployed in *AS1* sees all the c2p links, but only one out of the three p2p links. Because of the routing policies, no route using links 3—4, or 5—6 is exported to *AS1*. Let now suppose that the VP is deployed in *AS5* (Fig. 1.5b). This VP sees all the p2p links but misses two out of the four c2p links because, although there exists a valley-free path using these links, none of them is the best path, they are thus not exported. To uncover

more routing information, researchers usually combine the different VPs. For instance, combining the partial views of *AS1* and *AS5* enables observing all the AS links in the topology. However, while the partial views of distinct VPs may differ, they often exhibit some redundancies. For instance on Fig. 1.5c, combining the partial view of *AS1* and *AS2* does not enable to see any additional AS link.

Strategies to improve visibility of VPs. Two possible ways have been studied to address the visibility limitation of the BGP VPs: *(i)* enabling add-path [104] for route collectors and *(ii)* enabling BGP Monitoring Protocol (BMP) [105] for route collectors [106]. Add-path is a BGP router capability negotiated during the BGP session establishment. This feature enables a BGP router to send not only its best route but its n best routes to the neighboring ASes. Enabling this feature on route collectors may increase the VP visibility over the Internet routing. Similarly, BMP provides additional information about the received routes. It consists of regular dumps of the routes received by all the peers, rather than only the best one. Although these two features enable discovering more information about the Internet routing ecosystem, they only allow to see the routes filtered by the best path selection process, not those missing because of the configured routing policies. In addition, a VP will only collect the best route of each of its peers (since its peers may not configure add-path or BMP), rather than all the possible routes, limiting the visibility over backup links.

Most of the BGP-related studies may be impacted by the gap in visibility due to the limited coverage of the actual BGP collector infrastructures. There is a real need today to study what are the challenges of the current BGP collector infrastructure and their limitation and reevaluate the way we collect BGP routing data.

2

Internet routing measurement ecosystem

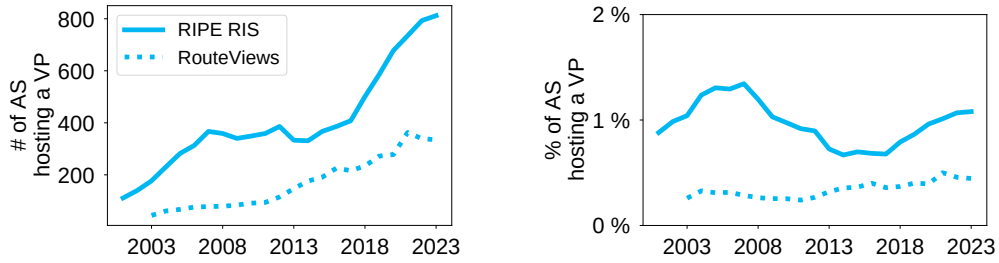
Although BGP data collection platforms being extensively leveraged to conduct a wide range of measurement studies about the Internet routing ecosystem, they face two conflicting realities that limit their effectiveness. *(i)* the current systems' visibility over BGP ecosystem is too low, resulting in a significant portion of the routing events being off the radar, therefore requiring a drastic expansion to overcome this problem; *(ii)* the volume of collected data poses an operational challenge to further expand these platforms. These limitations result in a less representative view of the real Internet, thereby impacting the results of many of the measurement studies.

In this chapter, we start in §2.1 by illustrating the limited visibility of the current RIS and RV infrastructures and demonstrating its impact on three popular measurement studies. Next, in §2.2, we identify an operational obstacle to the expansion of BGP data collection platforms, the volume of collected data. Finally, we present in §2.3 the challenges of scaling these platforms for both their operators and researchers.

2.1 The limited visibility of the BGP routing ecosystem

In the previous chapter, we explained why each BGP VP only has a local view of the Internet topology. Although combining multiple VPs enables discovering significantly

more topology, some portions of the Internet can only be observed locally, necessitating the deployment of a VP in these regions for analysis. However, the percentage of ASes hosting a VP remains low, even with the continuous expansion of platforms like RIS and RV.



(a) Number of VP deployed over time. **(b) % of VP deployed over time.**

Figure 2.1: Development of RIPE RIS and RouteViews data collection infrastructures.

2.1.1 RIS and RV topology coverage is low

Over the years, RIS and RV have deployed a substantial number of VPs across numerous ASes worldwide. Fig. 2.1a, presents the evolution of the number of ASes hosting at least one VP for both BGP collection platforms. In ten years, RIS doubled its *net coverage*, i.e., the number of ASes where a VP is deployed, increasing from 333 ASes in 2013 to 817 in 2023. Similarly, RV expanded its net coverage from 147 ASes in 2013 to 335 in 2023. Although this represents significant improvements in Internet topology net coverage, the expansion of these platforms does not overtake the growth of the Internet topology. In 2013, there were 45k ASes operating in the routing system, a number that increased to 75k by 2023 [16]. Consequently, the proportion of ASes hosting at least one VP has remained flat over time. As illustrated in Fig. 2.1b, the *VP coverage* — defined as the proportion of ASes hosting at least one VP — has never exceeded 2% (RIS and RV combined) over the two decades of these platforms' existence. Even when considering solely the *transit* ASes, i.e., the ASes that have at least one customer, the VP coverage remains low at 5.9% in 2023. This limited VP coverage leaves significant portions of the routing ecosystem off the radar, providing opportunities for attackers to evade detection by monitoring systems.

2.1.2 Attackers can escape from route collectors

BGP monitoring platforms rely on BGP data collected by the BGP data collection platforms. Given the current low VP coverage, many routing attacks may evade detection by VPs and consequently remain undetected by monitoring tools. For instance,

in Fig. 1.3a, the MOAS hijack intended by *AS8* cannot be detected without a VP either in *AS4* or *AS7*, even though the attacker does not attempt to evade BGP collectors. Milolidakis et al. demonstrated the feasibility of routing attacks without detection by BGP collectors [5]. The first step for the attacker is to identify the *dangerous monitors*, i.e., the VPs likely to detect the attack. They determine these VPs based on the proximity of a VP to the victim compared to its proximity to the attacker. Once the attacker identifies which AS hosting a VP is susceptible to detecting the hijack, they manipulate the AS path to exploit a BGP feature, the loop detection. Upon receiving a new route, a BGP router checks for loops in the AS path, i.e., it verifies if its ASN is already present in the AS path, in which case the route is dropped. By prepending the ASN of the dangerous monitors, an attacker can ensure that the route never propagates to a route collector. Additionally, according to [5], the attacker can use AS-path prepending to make the route less attractive, thereby reducing the likelihood of its propagation to a dangerous monitor. They also demonstrate that hijackers can perform effective attacks even if the VP coverage was four times higher.

The feasibility of this type of attack underscores the urgent need for a drastic increase in VP coverage. We demonstrate how expanding the BGP data collection platforms would improve the accuracy and coverage of scientific and operational analysis of Internet infrastructure.

2.1.3 Estimating the gap of visibility in the Internet routing

Measuring the gap of visibility is currently not a feasible task due to the lack of ground truth. While we cannot know how much additional information we might observe from VPs not peering with the public collection platforms, we estimate this gap using controlled simulations. We use C-BGP [107] to simulate "mini" Internets where each AS runs one BGP router and announces one or more prefixes.

Experiment settings. We run our simulations of AS topologies with 6k nodes (or 1k depending on the objective). Although scaling our simulations to match the size of the Internet is impractical, we note that, to the best of our knowledge, they are larger than simulations conducted in previous studies [26, 50, 108]. We assume that every AS in our simulation operates a single BGP router and we ensure that the number of prefixes announced by the ASes follows the distribution observed in the real Internet. We run our simulations on AS topologies generated using two different techniques.

Pruned known AS topology: We derive the AS topology from CAIDA's AS relationship dataset from October 2023 [109]. To reduce computational cost (in terms of hardware resources), we prune the topology by iteratively removing the leaf nodes and their corresponding connections until the topology has the required size (6k or 1k nodes

depending on the objective). We configured the routing policies to follow the Gao-Rexford model [19]. We run our simulations on one topology built using this process, as it does not induce any random behavior.

Artificial topologies: We use the hyperbolic graph generator [110] to build an AS topology whose parameters match those observed in the real Internet. We set the average node degree to 6.1, as observed in [109] and we use a power law with exponent 2.1 (as in [110]) for the degree distribution. The Tier1 ASes, i.e., ASes at the top of the Internet topology with no provider, are the three ASes with the largest node degree and are fully meshed. ASes connected to Tier1 are one level below in the Internet hierarchy, known as Tier2. ASes connected to a Tier2 but not a Tier1 are Tier3, and so forth. The AS relationships are defined as follows: two ASes have a p2p relationship if they are on the same level and a c2p relationship otherwise. We configure the routing policies to follow the Gao-Rexford model [19]. We run our simulations on ten topologies built using this process, using a different random seed for each.

Studied objectives. We leverage the generated topologies to evaluate the impact of VP coverage on our ability to conduct three canonical measurement studies: forged-origin hijack detection, link failure localization, and AS topology mapping.

I *Forged-origin hijack detection*: For every possible victim in our scenario, we simulate one Type-1 and one Type-2 hijack. A Type-X hijack denotes a forged-origin hijack where the attacker appears at position X in the AS path. The attackers are randomly selected and hijack one of their victim's prefixes.

II *Link failure localization*: For each run of our simulation, we create 1k random link failures and measure how well they can be localized using data from a given set of VPs. We localize the failures using the algorithm proposed by Feldmann et al. [111] which leverages the collected AS paths. We consider failures on p2p and c2p links separately, since routing policies typically reduce the propagation of p2p links, rendering them harder to observe. As the localization algorithm is computationally expensive, we use a topology of 1k ASes, instead of the 6k for the other objectives.

III *AS topology mapping*: We measure the proportion of AS links that can be observed with a given set of VPs. We collect the AS paths observed from these VPs and extract all the AS links. Similarly to objective *II*, we evaluate c2p and p2p links separately.

Simulation of VP deployment. These simulations aim at demonstrating the impact of the VP coverage on our ability to perform the three measurement analysis described above. We simulate the VP coverage by using data from a limited proportion of ASes, ranging from 0.5% to 100%. We iteratively select new ASes (randomly) until we use

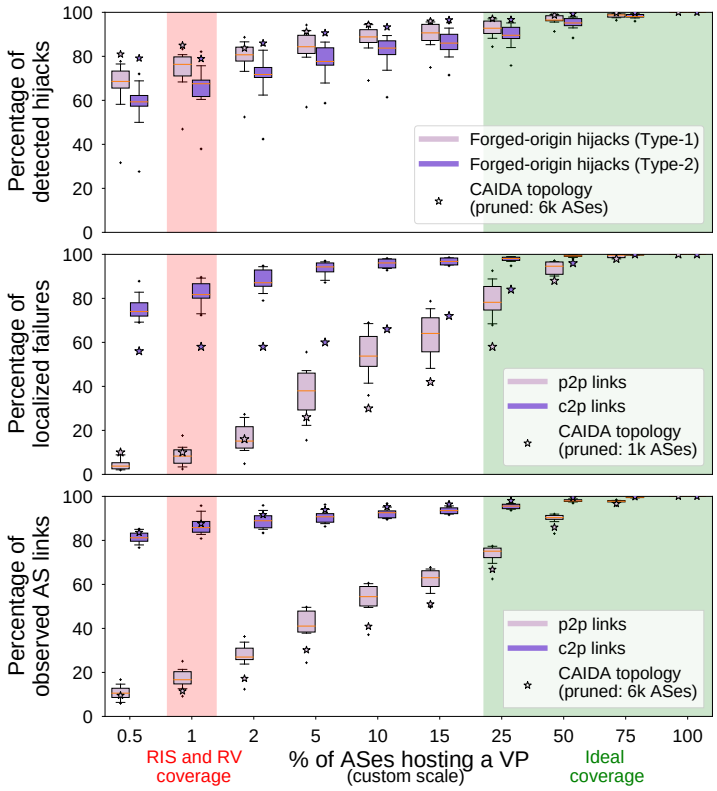


Figure 2.2: Our simulations show that the current RIS and RV coverage (1.1%, the red area) induces a significant impairment to important operational analyses. We suggest a 25-100x higher coverage (green area).

data from all the ASes. We then present in Fig. 2.2 how well we achieve an objective according to the percentage of ASes selected in our set of VPs.

Simulation results. Fig. 2.2 shows the percentage of detected hijacks (top), localized failures (middle), and observed AS links (bottom) as a function of the VP coverage, i.e., the percentage of ASes hosting a VP. The results of the simulations with the pruned topology are indicated with a star whereas the results from the ten artificial topologies are shown in the boxes. The red area on the graphs represents actual RIS and RV (combined) VP coverage. The results enable us to make two key observations.

Key observation #1: The simulations effectively illustrate the gap in the visibility induced by a VP coverage of 1%. We observed that, with the current RIS and RV coverage, we are not capable of performing accurately in any of the three objectives.

I Forged-origin hijack detection: With a VP coverage of 1%, we fail to detect 24% of the Type-1 hijacks (in the median case) when considering the artificial topologies, i.e., 24% of the Type-1 hijacks are invisible from *every* VP among the selected 1%. This number is 16% when considering the pruned topology. Unsurprisingly, Type-2 hijacks are even less visible (32% of undetected hijacks in the median case for the artificial topologies), since the AS path is longer resulting in a less attractive route that propagates less. The implication is that forged-origin hijack detection systems [69, 82] miss a significant fraction of the hijacks, even when using all RIS and RV VPs. Given the prevalent use of these platforms for hijack detection, their lack of coverage leaves significant attack surfaces open [5].

II Failure localization: With the current VP coverage of RIS and RV, only 10% (median) of the failures on p2p links can be accurately localized when considering either the artificial or pruned topologies. 20% of the failures on c2p links cannot be localized, a significant difference with p2p links, since p2p links are more challenging to observe than c2p links.

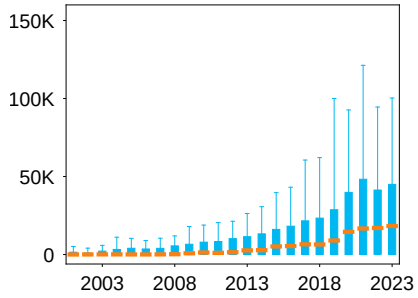
III AS topology mapping: With 1% of the ASes hosting a VP, our simulations show that we are able to observe only 16% of the p2p links in the median case considering the artificial topologies. With the pruned topology, we observe even fewer p2p links, amounting to 12%. Similarly to failure localization, p2p links are significantly more challenging to observe since the routing policies reduce their propagation.

Key observation #2: Our simulations suggest that the VP coverage should grow by at least 25x to achieve the three objectives reasonably well. According to our simulations, with a VP coverage of 50%, we may miss only 4% of the forged-origin hijacks, six times less compared to the actual VP coverage. We may be able to accurately localize 95% of the failures on p2p links (9 times more than today) and observe 90% of the p2p links (8 times more than today).

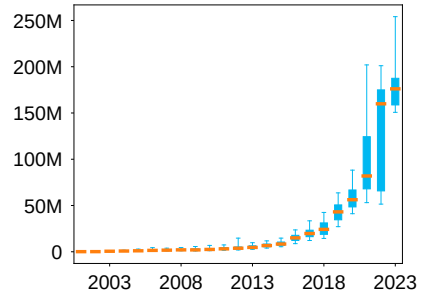
We tried to confirm these results with real but private data. We contacted a private BGP data provider, bgp.tools [103]. We compared the set of AS links observed by this provider with the set of AS links observed by RIS and RV in September 2023. We found that bgp.tools saw 192k AS links that none of the RIS and RV VPs observed, and conversely, RIS and RV VPs observed 401k links that bgp.tools did not observe. Other private data collection systems, e.g., that companies support, have reported visibility not seen in the public systems [112]. These significant differences in visibility across VPs provide compelling motivation to increase the VP coverage of public BGP route collector platforms.

2.2 The increasing volume of collected data

The main non-technical challenge that prevents BGP data collection platform expansion is the volume of collected data. We propose a measurement analysis demonstrating the substantial surge in the volume of collected data by the 1537 VPs from RIS and the 1130 VPs from RV.



(a) Volume of collected data per VP per hour over years.



(b) Volume of collected data per hour by all VPs combined over years.

Figure 2.3: Increase of the volume of collected BGP data over time.

The volume of data collected per VP increased. As discussed in the previous section, in the last two decades the Internet has experienced substantial growth. The number of ASes participating in the Internet went from 16k in 2003 to 75k in 2023 [16]. Similarly, the number of announced prefixes increased significantly, going from 120k in 2003 to more than 1M in 2023 [16]. This expansion of the Internet routing ecosystem induced a significant increase in the volume of data collected per VP. Elmokashfi et Al. demonstrated that the growth in the number of collected updates per VP follows the growth of the Internet [113]. We demonstrate this by selecting all the VPs that have existed since 2001 and that remain active today. We downloaded all the collected BGP updates from these VPs since 2001. To reduce the computational expense of this analysis, we focused on one hour per day, randomly selected (the same for every VPs). We present the results of this analysis in Fig. 2.3a. The X-axis represents the time and the Y-axis represents the number of BGP updates collected per hour by one VP. We can clearly observe that the volume of data collected per VP has continuously increased over the years, reaching 26k updates collected per hour per VP (median case) in 2023.

The volume of data collected by RIS and RV increased in a quadratical fashion. The compound effect of both a (approximately) linear increase in the number of deployed VPs and the linear increase in the volume of data collected per VP is a quadratic increase in the volume of collected data by RIS and RV. We demonstrate this by collecting all the BGP updates from all RIS and RV data since 2001. Similarly to the

results of Fig. 2.3a, we restrict our analysis to only one hour of data (randomly selected) per day to manage computational expenses. We present the results of this experiment on Fig. 2.3b, where the X-axis represents the time and the Y-axis represents the number of collected BGP updates per hour by all the VPs. As expected, we observe a quadratic increase in the volume of collected data, going from 561k in 2003 to 184M in 2023. This number of BGP updates represents approximately 1TB of data collected per day by the RIS and RV infrastructures, encompassing both RIBs and raw BGP updates (before any lossless compression strategy).

The increase in the volume of data has been a prohibitive factor for using the BGP data by the users, but is also a challenge for the collection platform operators to maintain.

2.3 The challenge of scaling BGP data collection platforms

Putting aside the non-technical challenges of a radical increase in the VP coverage, we focus first on the technical challenges, for both data providers and users.

2.3.1 Challenges for data providers

Cultivating more VPs generates more data as each of them exports BGP updates and leads, as we shown in §2.2 to a *quadratic* increase of the collected volume of data. RIPE RIS operators have expressed some concerns about the volume of data they collect [114]. Consequently, they had to revise their peering policies, by accepting only new VPs upon request. However, even without chasing themselves for interesting VPs, they are currently working on strategies to focus only on the most useful new BGP VPs [115] and have adopted a *selective peering* strategy for almost three years now. This selective peering strategy is also used by bgp.tools [103], which uses the number of newly discovered AS links to evaluate the relevance of a new BGP VP. The volume of data has been so important that RIS started to question its data retention principles [116]. The BGP data takes around 800TB of disk space for RIS, with 80% accounting for the last 5 years. They proposed different strategies to cope with the storage and distribution costs, such as using cold storage for rarely accessed data, or limiting the data collection rate for non-authenticated users.

2.3.2 Challenges for users

Although several tools can speed up data processing [86, 87, 117], many measurement studies and monitoring tools use only a subset of the available data collected by RIS and RV. They either use a subset of the VPs or reduce the timespan of their analysis. While authors of these studies do not typically explain why they sample, their choice suggests they believe the data volume is not worth trying to manage. We confirm this

explanation with a survey we conducted on authors of eleven research papers. We do not cite any of them, as we ensured them to preserve their anonymity.

We classified eleven BGP-based studies, selected from top computer network and system conferences (SIGCOMM, NSDI, S&P, USENIX Sec, NDSS, and IMC) based on their BGP data sampling strategy. We selected only *recent* analysis, (2017 for the oldest) that are focused on *different* objectives. For instance, we did not select two different analysis aiming at studying RPKI deployment. Additionally, papers involved in any of the use cases studied in the evaluation (§6) are omitted. Nine papers used **all routes** collected from a **subset** of the available VPs. These papers are classified in category C_1. Six papers **limited the duration** of their experiences, which corresponds to category C_2. Note that the categories can overlap. We emailed the authors for all these studies and asked them whether the volume of BGP data was a limiting factor in their studies, how and why they sampled BGP data sources, their understanding of the impact of the sampling on the quality of their results, and if they would expand their sample given more resources or time. We did not receive any response for three of the eleven papers. Thus, we eventually collected feedback from seven respondents in C_1 and five in C_2.

Detailed answers. Table 1 lists the questions we asked the participants of our survey along with their detailed answers. We color the answers based on our interpretation of whether the responses reflect some concerns about the volume of the BGP data (green) or not (red). Neutral answers are colored in blue.

Key observations. We interpret the results of the survey and make two key observations.

Key observation #1: The volume of BGP data to process is often a limiting factor. In fact, seven (of eight) respondents found the BGP data expensive to process. For three respondents in C_1, processing time motivated them to use only a subset of the VPs; three respondents in C_2 considered the processing time when choosing a measurement interval. Even a respondent who used a Spark cluster found it prohibitively time-consuming to process the BGP data.

Key observation #2: Users often sacrifice the quality of the results to facilitate the data processing. In fact, six respondents in C_1 acknowledged that using more VPs would improve the quality of their analysis. The last respondent was not sure, given the potential redundancy in the data sources (which he did not analyze). Two of the six believed it would not significantly change the conclusion of their studies (e.g., one said that it could help to pinpoint corner cases). However, six of the seven authors in C_1 affirmed that they would have used more VPs given additional resources and time. Similarly, all five respondents in C_2 said that extending the duration of their study would improve the quality of their results. One respondent thought the gain would not

Collection strategy	Questions asked	Collected answers
C_1: **All routes and subset of VPs (seven papers)**	Why did you use a subset of the VPs ?	**To speed up data processing (x2)** **For disk space and time efficiency (x1)** **I thought the rest would be similar (x1)** I did not manage to use them all (x2)
	How did you select your VPs ?	**I took them randomly (x2)** **I do not remember (x2)** It was arbitrary: my script partially failed (x1) I took geographically distant BGP collectors (x1) I did not manage to use VPs from one data provider (x1)
	Do you think more VPs would improve the quality of your results?	**Yes (x4)** Results would be similar, but it can help to find corner cases (x1) Yes, but not significantly (x1) I am not sure (x1)
	Would you have used more VPs if you could?	**Yes (x4)** **Yes, I'd love to (x1)** **Definitely (x1)** I am not sure, but I don't think so (x1)
C_2: **Limited duration of experiment (five papers)**	Was the processing time a factor that you considered when you decided on the duration of your measurement study?	**Yes (x3)**
	Do you think extending the duration of your measurement study would improve the quality of your results?	**Yes (x2)** **Yes, especially for rare events (x1)** Potentially (x1) Yes, but not significantly (x1)
	Would have extended the duration of your measurement study if you had more resources?	**Yes (x2)** Yes, but it depends on the time remaining before the deadline (x1) I think so, but also if I had more time before the deadline (x1)
All eight papers	Do you find the data from RIS and RouteViews expensive to process in terms of computational resources?	**Yes (x1)** **Yes, CPU and storage (x2)** **Yes, the storage cost and the download cost are very large (x1)** **CPU is the main issue (x1)** **RIS data takes a lot of time to download, especially when we need data for multiple days (x1)** Not the worst, but we definitely need a resourceful server if we want to catch some deadline (x1) We did that in a server so that was not a huge issue (x1) **No (x1)**
	Is there any additional challenge that you encountered when processing the BGP data from RIS and RouteViews?	**Our team used Spark clusters and Python but it was too slow (x1)** **We had to download the data from all VPs as there is no optimal solution for selecting them, the storage overhead and time overhead were very high (x1)** **It'll be helpful to make processing faster less resource-consuming (x1)** Too many duplicate announcements make processing harder (x1) Variable sizes of update files exacerbate scheduling parallelization (x1) RIS took a lot longer than RouteViews (x1) We had issues when collecting updates in real-time (x1) We had to deal with bugs in BGPdump (x1) Broken data feeds and data cleanup is also an issue that we need to take care of (x1)

Table 1: Details of the questions asked to the participant and their answers.

be significant; another said it could help detect rare routing events. All respondents in C_2 would have extended the duration of their observation window given more time and resources. The uncomfortable truth is that we do not know exactly what they are missing, which is why we used simulations and experiments to corroborate that important analyses lose accuracy and/or coverage when using heavily sampled topologies.

Common BGP data sampling schemes. Among the seven respondents who took the data from a subset of the VPs, one picked geographically distant VPs. While intuitive, this strategy fails to optimize for some metrics (e.g., AS link coverage, see §6). Another respondent unintentionally removed some VPs (leaving an arbitrarily selected set in the study) and two did not remember how they selected their VPs. All the remaining

respondent selected their VPs arbitrarily. We show in the benchmark (§6) that sampling data in an unoptimized fashion, i.e., arbitrarily or with simple metrics leads to poor performance for most of the use cases.

Ethics of our survey. The participants of the survey freely participated. We contacted them by email to ask them whether they would agree to participate. We stated the purpose of the survey and notified them we might publish the results anonymously. Following is the exact wording we used when soliciting participants.

"I would like to know whether you would be willing to answer a quick survey about why you selected these VPs and the impact that you think this selection made on your measurement study. Answering this survey will help us to better understand how researchers proceed when selecting BGP vantage points, why they often do not take them all, and what is the impact of the vantage points selection on the results of the measurement studies. The survey includes a few questions that I will send you by email if you agree to answer them. It should take less than 5 minutes to answer it. We might publish the results of our survey. If we do that, we will either do it in a manner that would not allow identification of your personal identity or we will ask your permission."

These different responses highlight the urgent need to improve the BGP data sampling strategies. One characteristic of the collected BGP data is its prominent redundant aspect. While combining the view of multiple VPs often enables discovering new information about the Internet routing ecosystem, the partial views of two VPs almost always exhibit a considerable rate of redundancy.

3

Redundancy in BGP data

Due to the prohibitive volume of BGP data to process, users resort to often arbitrarily sampling the data. Although sampling BGP data can lead to a loss of visibility over Internet routing ecosystem, there is an opportunity to minimize information loss while significantly reducing the volume of data to process: leveraging BGP data redundancy. The Internet hierarchical structure, where a few ASes provide transit for a large portion of the Internet, concentrates the connectivity toward these ASes. As a result, VPs have a good view of the core of the Internet, leading to substantial redundancies in their partial views.

In this chapter, we start in §3.1 by illustrating through an example how can BGP data be redundant. Then we propose in §3.2 a characterization of the redundancies in the BGP data.

3.1 Illustrating BGP data redundancies

In this section, we illustrate BGP data redundancies using a simple example. Fig. 3.1 depicts a small Internet topology with eight ASes. Arrows represent a c2p relationship while lines represent a p2p relationship. Routing policies are configured according to the Gao-Rexford model. *AS4* announces two prefixes p1 and p2 and *AS6* announces

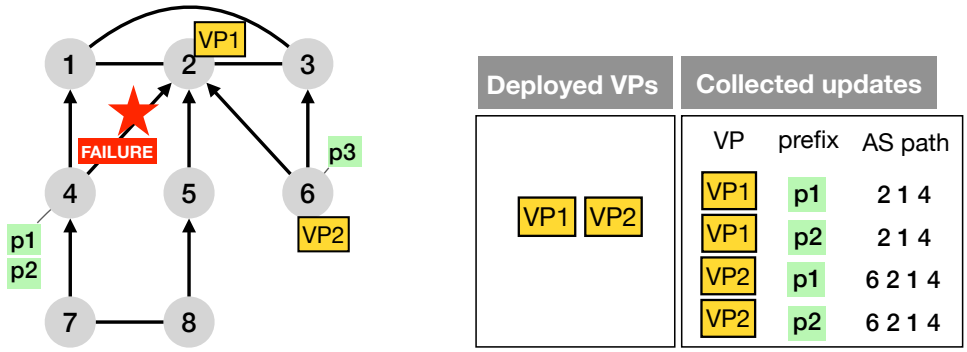


Figure 3.1: Example of possible redundancy among BGP data.

p3. In this scenario, two VPs are deployed, one in *AS2* and one in *AS6*. Both VPs export all their best routes to RIS and RV. We consider a BGP event where the link 4—2 fails.

Since the best path from *AS6* and *AS2* to p1 and p2 uses link 4—2, the failure on this link will trigger new announcements with an alternative AS path that circumvents the failure. On the right of Fig. 3.1, we show the BGP updates collected by VP1 and VP2. Both VPs receive two new routes: one for p1 and another for p2. Unsurprisingly, we can observe strong data redundancies, at two different levels.

Redundancy between updates: As p1 and p2 are announced by the same AS and are not influenced by traffic engineering decisions, they follow the same path propagation. Consequently, both VP1 and VP2 receive two BGP updates with the same AS path, one for each of the two prefixes. These two BGP updates are received at the same time and are triggered by the same event, indicating a high level of redundancy.

Redundancy between VPs: VP1 and VP2 provide a very similar view of the Internet topology. The path toward p1 and p2 received by VP1 is included in the path received by VP2. In addition, since VP1 and VP2 are close in the topology, they receive these two paths at roughly the same time. All these similarities highlight the high level of redundancy between these two VPs.

While this section only aims at providing an intuition on how BGP data can be redundant, we now present a rigorous measurement analysis to characterize redundancies in the BGP data.

3.2 Exploring BGP data redundancies

We now present a comprehensive analysis of redundancy in the BGP data. As there is no consensus on how redundancy must be defined, we propose three gradually stricter

definitions of redundancy between updates. We also use these definitions to define redundancy between VPs.

We denote as $u(vp, t, p, L, L_w, C, C_w)$ a BGP update collected by VP vp at time t for prefix p. L corresponds to the set of AS links in the AS path of the BGP update, while L_w represents the set of AS links implicitly withdrawn by the new AS path, i.e., the set of links that were part of the previous AS path for prefix p, but not part of the new one. Similarly, C is the set of communities announced by u and C_w is the set of communities implicitly withdrawn for prefix p. Observe that $L_w = C_w = \emptyset$ if there was no previous update for p observed by vp or if the new update for p is the same as the new one (except for the timestamp). Considering two BGP updates $u1(vp1, t1, p1, L1, L1_w, C1, C1_w)$ and $u2(vp2, t2, p2, L2, L2_w, C2, C2_w)$, we propose the following definition of redundancy:

Definition 1 (prefix based). *Update $u1$ is redundant with update $u2$ if the two following conditions hold:*

- $|t1 - t2| < 100s$
- $p1 = p2$

The first condition uses a 100-second slack when comparing the two timestamps to accommodate typical BGP convergence time [42]. The second condition checks if the two updates involve the same prefixes. This definition is useful for listing all prefixes routed on the Internet. By eliminating BGP updates with the same prefixes, one is capable of reducing the volume of collected data without missing any prefix.

Definition 2 (prefix and AS-path based). *Update $u1$ is redundant with update $u2$ if the three following conditions hold:*

- $|t1 - t2| < 100s$
- $p1 = p2$
- $L1 \setminus L1_w \subset L2 \setminus L2_w$

This definition introduces a third condition that checks whether the set of new links in the AS path observed by $vp1$ is included in the set of new links in the AS path observed by $vp2$. This definition is valuable for AS-level topology-related measurements. For example, when mapping the Internet topology, reducing duplicated links helps minimize the number of processed updates without missing any AS links. Having more diverse AS paths may also be useful for inferring the AS relationships.

Definition 3 (prefix, AS-path and community based). *Update $u1$ is redundant with update $u2$ if the four following conditions hold:*

- $|t1 - t2| < 100s$
- $p1 = p2$
- $L1 \setminus L1_w \subset L2 \setminus L2_w$
- $C1 \setminus C1_w \subset C2 \setminus C2_w$

This final definition adds a fourth condition involving BGP communities, checking whether the set of new BGP communities announced by the update received by $vp1$ is included in the set of new BGP communities observed by the update collected by $vp2$. This definition is particularly useful for detecting new BGP communities, as it enables the collection of a wide range of BGP communities while keeping the number of collected updates manageable. Note that both Def. 2 and Def. 3 are asymmetric, as for two sets X and Y, $X \subset Y \implies Y \subset X$.

Despite their simplicity, these definitions enable us to provide a comprehensive analysis of the redundancy in BGP data. It is important to note that these definitions are not used in the design of any of our systems. To evaluate the BGP data redundancy, we collect all BGP updates from RIS and RV during the first two hours of September 2023 and apply our three definitions to the collected data.

The vast majority of the collected updates are redundant with another collected update. For each BGP update collected by RIS and RV, we evaluate the proportion of redundant updates, according to our three definitions. We find that 97% of the updates are redundant with at least one update collected by another VP, according to Def. 1. Unsurprisingly this number decreases with stricter definitions but remains high: 77% with Def. 2 and 70% with Def. 3.

A significant portion of the VPs are redundant with another VP. We use our update-wise redundancy definition to quantify redundancy between RIS and RV VPs. We define $vp1$ as redundant with $vp2$ if at least 90% of $vp1$'s updates are redundant with one or more updates from $vp2$. We randomly select 100 VPs from RIS and RV and present the proportion of redundant updates for each pair. To reduce potential biases induced by a VP selection, we run this experiment 30 times with different sets of VPs and present the median results (in terms of the proportion of redundant updates). We show in Fig. 3.2 the results of this experiment for the three gradually stricter definitions.

Fig. 3.2 represents three heatmaps, each corresponding to one redundancy definition. In these heatmaps, a cell indicates the redundancy level between the VP in the X-axis and the VP in the Y-axis. The cells are colored in red if the proportion of redundant updates is above 90%. Note that heatmaps are not symmetric due to the asymmetric characteristic of Def. 2 and Def. 3.

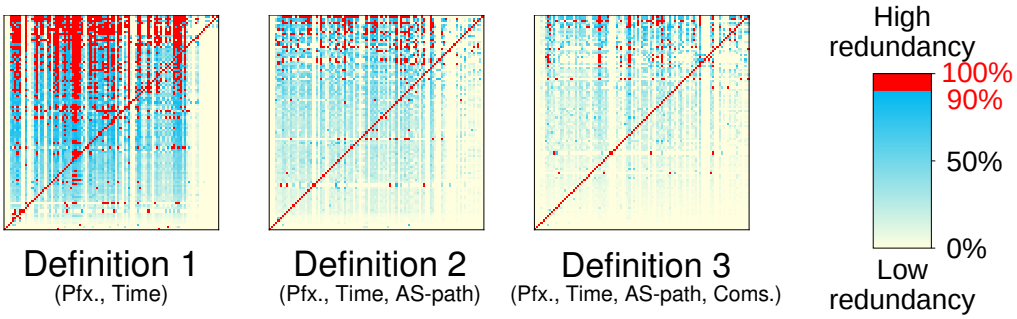


Figure 3.2: Redundancy among 100 random RIS and RV VPs for three gradually stricter redundancy definitions.

With Def. 1, 70% of the VPs are redundant with at least one other VP. Unsurprisingly, this number decreases with stricter definitions. With Def. 2 (resp. Def. 3), 26% (resp. 22%) of the VPs are redundant with another one. Some VPs do not export a *full feed*, meaning that they are not sending a route for all the prefixes on the Internet. The results of this experiment remain consistent when considering only full feeders.

This analysis highlights the high level of redundancy in the BGP data. This redundancy presents an opportunity for sampling BGP data without losing information about the Internet routing ecosystem. Although these three definitions enable us to illustrate the redundancy among BGP data, none are used in the design of any of our systems. Instead, our sampling algorithms rely on data-driven strategies that prevent overfitting to any specific objective. This feature enables our sampling to remain efficient, regardless of what the user wants to do with the data.

4

Sampling strategies

The redundancy in the BGP data offers an opportunity for sampling, enabling a significant reduction of the data to process, without losing valuable information. In §3.2, we explored the redundancy in BGP data using three simple definitions. Although these definitions effectively demonstrate the high level of redundancy in BGP data, they are too simplistic to provide a general sampling methodology that works regardless of the user's objective. Generally, optimizing our algorithms based on a specific definition of redundancy results in overfitting.

We explore this risk of overfitting by designing three *definition-based specific* baselines, each optimized for one of the three redundancy definitions presented in §3.1.
Definition-based Specific baseline: A definition-based specific baseline minimizes the redundancy among the BGP updates collected by a set of VPs according to a given definition of redundancy. More precisely, it selects VPs by greedily adding the non-selected VP that less increases the redundancy among the BGP updates collected by the selected VPs, according to the redundancy definition it optimizes.
Unsurprisingly, using these baselines instead of a random selection of VPs significantly lowers the number of VPs that are redundant with at least one other. With Def. 1, this number is 37, almost twice as less as with a random selection. This number decreases with stricter definitions, 20 with Def. 2 and 15 with Def. 3. While this approach

effectively reduces the redundancy according to a given definition, we show in the evaluation (§6) that it performs poorly for other objectives.

Although they do not overfit, naive or arbitrary sampling fails to work for a various and large range of objectives. When selecting dissimilar VPs to sample BGP data, a common strategy is to select VPs geographically distant or distant in terms of AS hops. Intuitively, VPs positioned at different locations on the Internet topology may observe different events, and thus provide more diverse information compared to VPs that are close within the Internet topology. We experimentally show that this strategy fails for a simple and common objective: mapping the Internet topology.

We collect the first RIB dump from all VPs on December 1, 2023, and construct the AS topology of each partial view by leveraging the AS paths extracted from the RIB. For each pair of VPs, we compute the proportion of links observed by only one of the two VPs, referred to as the proportion of unique links. We present in Fig. 4.1 the proportion of unique links as a function of the distance between the two VPs (in terms of AS hops).

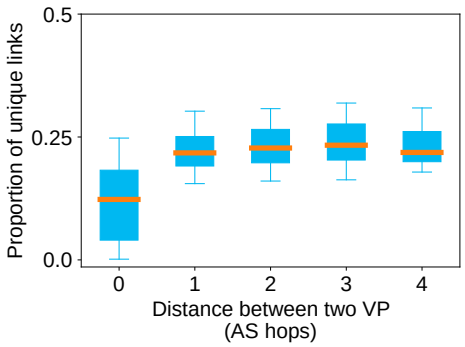


Figure 4.1: Proportion of unique AS links seen by a pair of VPs as a function of the distance between two VPs.

Unsurprisingly, the higher the distance between the VPs, the higher the proportion of unique observed links. However, while this tends to be true most of the time, it does not generalize. For instance, in the median case, two VPs deployed in the same AS have 12% of unique observed links. Additionally, in 47% of the cases, two VPs distant by four hops have a higher proportion of unique observed AS links than the median for two neighboring VPs. Using a metric such as geographic diversity [118] also yields poor results, as we show in the evaluation (§6).

Sampling BGP data based on specific definitions inevitably leads to undesirable overfitting effects. Sampling BGP data based on generic, but naive definitions of redundancy leads to poor performances for some popular objectives. Therefore, it is crucial to design efficient definitions of redundancy that perform well *regardless of the objective*.

We identified two levels of redundancies: redundancy between updates and redundancy between VPs. In this chapter, we present one efficient definition for each type. We start by proposing BUS (BGP Update Selector), an algorithm that evaluates the redundancy between updates (§4.1), based on a new metric that we designed called *reconstitution power*. Then, we present MVP (Most Valuable VP), an algorithm that defines the redundancy between VPs, based on the impact of carefully selected events on the partial view of each VP.

4.1 Sampling at the update granularity

To reduce the number of updates to process while minimizing information loss, a simple technique is to retain only one route among those that share the exact same BGP attributes, but with a different prefix. This strategy seems quite efficient since it reduces the number of BGP updates to process by $\approx 85\%$ (i.e., 6 times less data to process). However, this strategy exhibits two main drawbacks. First, it does not exploit the redundancies between the VPs, which are quite common as illustrated in Fig. 3.2. Second, it can be only applied to past updates and not to future updates, as there is no guarantee that two prefixes sharing the same attributes at a given time will also share the same attributes in the future. Therefore, we need to rely on a definition that addresses both these points.

In this section, we present an algorithm, BUS, that leverages a new metric to evaluate the redundancy between updates: the *reconstitution power*. This methodology is data-driven, leveraging *past updates* to determine the redundancy of *future updates*, as we observed that updates redundant at a given time are likely to remain redundant a short time after (see §4.1.1). Our metric enables identifying which updates can be discarded while minimizing information loss. BUS's update selection process involves three steps: *(i)* grouping the *correlated updates*, i.e., updates that are induced by the same BGP event, *(ii)* applying our new metric, the reconstitution power to identify which updates can be discarded, and *(iii)* exploiting the per-prefix redundancy to further reduce the number of retained BGP updates. We provide a summary of all notations used in this section at the end of the chapter, see Table 4.

4.1.1 Finding correlated updates

Any BGP event occurring on the Internet typically triggers new BGP route updates. For instance, a failure on an AS link will generate new announcements with AS paths that circumvent the failure. Similarly, a hijack will trigger the appearance and the propagation of route updates with the hijacked BGP prefix, and a routing policy change will trigger new announcements such that the subsequent paths accommodate the new policies. These new route updates will propagate and likely reach multiple VPs.

For instance, in Fig. 3.1, the failure on link 4—2 triggers route updates with a path circumventing the failure, and these announcements are collected by two distinct VPs. Therefore, updates triggered by the same events are likely to carry redundant information (similar AS paths on Fig. 3.1). While it is challenging to precisely identify which updates are triggered by the same events, these updates share two key similarities: *(i)* they all carry information about the same prefix and *(ii)* they are seen roughly at the same time (accommodating path exploration and convergence delays).

To capture the correlations between BGP updates, BUS builds *correlation groups*, which group the updates likely triggered by the same BGP event. A correlation group comprises all the updates that pertain to the same prefix and are observed roughly at the same time. Essentially, a time-correlated group of updates means that when one element of this group is observed, the others are likely to be observed after a short time. We now provide a formal definition of a correlation group.

Correlation group definition A correlation group is a set of updates where each update is represented by three information: the *VP* that collects the update, its *AS path*, and its *communities*. We denote an update collected by VP vp at time t with prefix p, AS path A, and BGP communities C as $u(t, p, vp, A, C)$. We denote its *representation* within a correlation group (vp, A, C) as $r(u) = (vp, A, C)$. Given two updates $u_1(t_1, p_1, vp_1, A_1, C_1)$ and $u_2(t_2, p_2, vp_2, A_2, C_2)$, their representation $r(u_1) = (vp_1, A_1, C_1)$ and $r(u_2) = (vp_2, A_2, C_2)$ are included in the same correlation group if u_1 and u_2 satisfy the following two conditions:

- $|t_1 - t_2| < 100s$
- $p_1 = p_2$

We choose a 100-second window to accommodate typical BGP convergence times [42]. If updates $\{u_0, \ldots, u_n\}$ are grouped into the same correlation group G, then $G = \{r(u_0), \ldots, r(u_n)\}$. Observe that in practice a correlation group contains significantly more than two updates, as an event can be seen globally, i.e., by nearly all the VPs (e.g., a new AS operating on the Internet). Using data from RIS and RV, the average size of a correlation group is 27. Since not all possible updates are triggered by a single BGP event, there are typically multiple correlation groups for each prefix. Distinct BGP events are likely to trigger different updates, resulting in distinct correlation groups.

Correlation group construction. To build correlation groups that accurately reflect the correlations between updates received by VPs, BUS uses two days of data collected from all the VPs. BUS then extracts all updates and groups them into the corresponding correlation groups. A significant proportion of the events that trigger BGP updates are recurrent, and correlations between past updates are likely to reappear in the future. For instance, routing changes due to traffic engineering occur regularly. In practice (i.e.,

using RIS and RV data), 96% of the updates that appear at a given time also appear with the exact same attributes (excluding the timestamp) during the following 24 hours. This recurrence leads to duplicate correlation groups, as similar events may generate similar correlated updates. Therefore, if two correlation groups are composed of the exact same update representations, BUS merges them into the same correlation group and increases the *correlation group weight* of this group, denoted as $w(G)$ for a correlation group G. More formally, two correlation groups G_1 and G_2 are merged into the same correlation group if they are equal, i.e., if the set of the update's representations that compose them are equal. Intuitively, the higher the correlation group weight of a group, the more correlated the updates within that group.

Correlation group construction time. BUS builds correlation groups using two days of BGP data collected from all the available VPs. The construction time must be long enough to ensure that correlation groups accurately represent the actual correlations between updates received by VPs. We experimentally show that two days is the optimal tradeoff between stability and computational expense. We tested values for this parameter ranging from one to ten, using ten different update periods to avoid biases induced by sampling BGP data over a short timeframe. We find that after two days, the ranking of correlation groups (in terms of correlation group weights) had a 94% probability of being the same as if we used another training period of the same length. This probability is 81% when using only one day of data, and 95.8% when using ten days. Increasing the construction time after two days results in marginal performance improvements, while significantly increasing the computational expenses.

Example of correlation group construction. We use the example depicted in Fig. 4.2 to illustrate how BUS builds the correlation groups. This example involves an eight-ASes Internet topology, where routing policies are configured according to the Gao-Rexford model. An arrow indicates a c2p relationship while a line represents a p2p relationship. Two VPs are deployed, one in *AS2* (VP1) and one in *AS6* (VP2), both of which export all their routes to route collectors. We consider four routing events, separated in time:

Event #1: The link 2—4 fails at time T_1.

Event #2: The link 2—4 is restored at time T_2.

Event #3: The links 2—4 and 2—6 fail at time T_3.

Event #4: The links 2—4 and 2—6 are restored at time T_4.

For all n in $\{1, \ldots, 3\}$, we have $T_{n+1} - T_n > 100s$. To simplify this example, we focus only on p1 and omit the BGP community, as it does not change the correlation group

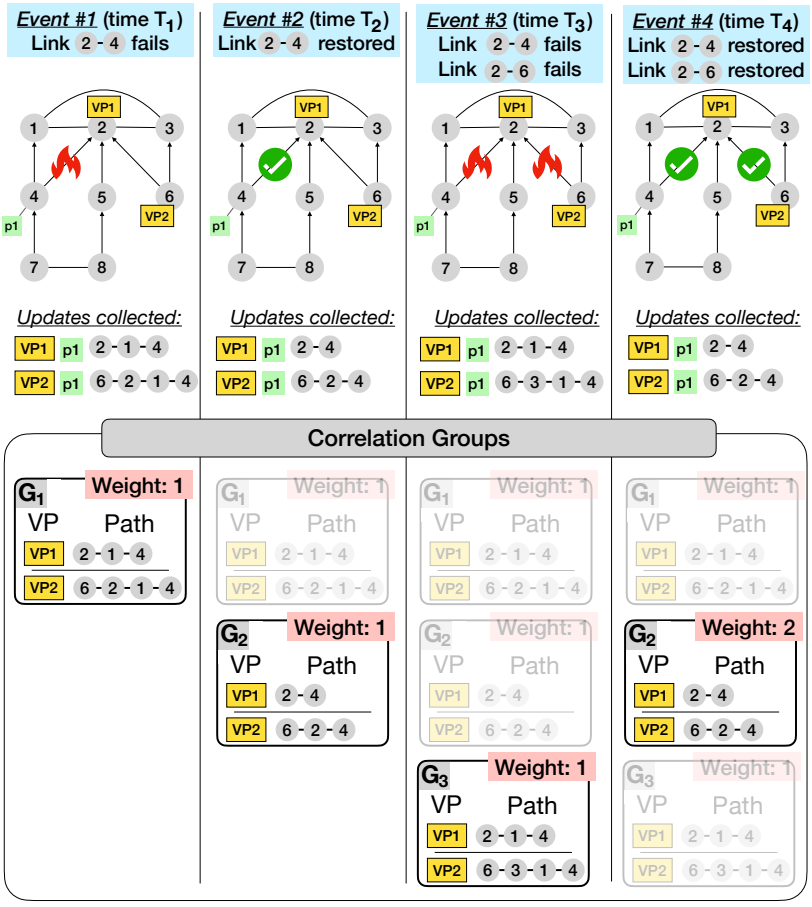


Figure 4.2: Example of how BUS builds correlation groups. We consider four BGP events occurring at distinct timestamps.

construction. The four events trigger eight different updates that we denote as u_i with $i \in \{1, \ldots, 8\}$:

u_1: seen at time T_1 by VP VP1 for prefix p1 with AS path 2—1—4

u_2: seen at time T_1 by VP VP2 for prefix p1 with AS path 6—2—1—4

u_3: seen at time T_2 by VP VP1 for prefix p1 with AS path 2—4

u_4: seen at time T_2 by VP VP2 for prefix p1 with AS path 6—2—4

u_5: seen at time T_3 by VP VP1 for prefix p1 with AS path 2—1—4

u_6: seen at time T_3 by VP VP2 for prefix p1 with AS path 6—3—1—4

u_7: seen at time T_4 by VP VP1 for prefix p1 with AS path 2—4

u_8: seen at time T_4 by VP VP2 for prefix p1 with AS path 6—2—4

We show how BUS groups these updates in correlation groups, such that they are representative of the real correlations between updates.

Upon T_1: The link 2—4 fails. The primary path (i.e., the best path) from *AS2* to p1 uses the failed link. *AS2* switches its best route to the alternative route 2—1—4, which circumvents the failure. This new route then propagates to *AS2* neighboring ASes, including *AS6* which adopts this new route. As a result, two updates are thus collected, u_1 and u_2. Both these updates share the same prefix and are assumed to be collected roughly at the same time since *AS2* and *AS6* are neighbors. Therefore, u_1 and u_2 thus satisfy the requirements to be grouped into the same correlation group. BUS creates a correlation group G_1, in which it adds both $r(u_1)$ and $r(u_2)$. Recall that $r(u)$ stands for the *representation* of u in a correlation group. We have $G_1 = \{($ VP1, 2—1—4$), ($ VP2, 6—2—1—4$)\}$ (as we omit BGP communities for the sake of simplicity) and $w(G_1) = 1$.

Upon T_2: The link 2—4 is restored. Since a better path now exists from *AS2* to p1 compared to the actual one (2—1—4), *AS2* switches its path to the new best one, 2—4. This new path propagates to *AS2* neighboring ASes, including *AS6* which adopt this route. As a result, two updates are collected, u_3 and u_4. Both these updates share the same prefix and are assumed to be collected roughly at the same time since *AS2* and *AS6* are neighbors. u_3 and u_4 satisfy the requirements to be grouped into the same correlation group. BUS creates a new group G_2 in which it adds both $r(u_3)$ and $r(u_4)$. We then have $G_2 = \{($ VP1, 2—4$), ($ VP2, 6—2—4$)\}$. Since G_2 is not equal to any of the existing groups, BUS does not merge it, and we have $w(G_2) = 1$.

Upon T_3: Two links fail, 2—4 and 2—6. Similarly to the failure upon T_1, the actual path from *AS2* to p1 uses a failed link, prompting *AS2* to switch its path to the one going through *AS1*. However, this new path is not propagated to *AS6*, since the link 2—6 also failed. Consequently, *AS6* uses a backup path going through *AS3*: 6—3—1—4. Two updates are collected, u_5 and u_6. Both these updates share the same prefix and are assumed to be seen roughly at the same time. BUS creates a new correlation group G_3 in which it adds both $r(u_5)$ and $r(u_6)$. Thus, we have $G_3 = \{($ VP1, 2—1—4$), ($ VP2, 6—3—1—4$)\}$. Despite having $r(u_5) \in G_1$, G_1 is not equal to G_3. Consequently, BUS does not merge G_1 and G_3, which results in $w(G_3) = 1$.

Upon T_4: Both links 2—4 and 2—6 are restored. With now all the links being restored, *AS2* and *AS6* can both use their primary best paths to reach prefix p1. Two updates are collected, u_7 and u_8. These updates can be grouped into the same correlation group, as they share the same prefix and are assumed to be received roughly at the same time. BUS creates a new correlation group, G_4, in which it adds both $r(u_7)$ and $r(u_8)$.

This results in $G_4 = \{($ VP1, 2—4$), ($ VP2, 6—2—4$)\}$. In this scenario, we have $r(u_3) = r(u_7)$ and $r(u_4) = r(u_8)$, which implies that $G_2 = G_4$. Consequently, BUS can merge G_4 into G_2, which results in $w(G_2) = 2$.

After the four events, BUS built three distinct correlation groups that capture the correlation between updates. We now explain how BUS uses these correlations to identify redundant updates using a new metric the *reconstitution power*.

4.1.2 Indentifying redundant updates

BUS identifies redundant updates using the constructed correlation groups (§4.1.1) and an update reconstitution algorithm that relies on a new metric that we designed called the *reconstitution power*.

Reconstitution power intuition: Given a set of BGP updates β, if it is possible to identically reconstitute β from one of its subsets α, then α contains the updates that carry the most valuable information, while $\beta \setminus \alpha$ contains redundant updates. An update is identically reconstituted if it is possible, using our reconstitution algorithm, to retrieve the VP it was collected from, the AS path, the community, the prefix,and the timestamp with a slack of 100 seconds (to accommodate convergence delays [42]).

The reconstitution power of a subset α of β corresponds to the proportion of updates in β that can be identically reconstituted from α. BUS's objective is to find a subset of updates α that maximizes the reconstitution power and discards the updates in $\beta \setminus \alpha$. BUS evaluates the reconstitution power of a set of updates using our *reconstitution algorithm*, which relies on the constructed correlation groups.

Reconstitution algorithm formalization. We denote as $Corr(p, u)$ the list of correlation groups for prefix p that include the representation $r(u)$ of update u. Recall that the representation of an update $u(t, p, vp, A, C)$ is (vp, A, C), where vp is the VP that collects u, A is its AS path and C is the set of associated BGP communities. We denote as $max_{weight}(\mathcal{G}, t, p)$ the function that takes as input a set $\mathcal{G}$ of correlation groups, returns the update representations included in the correlation group in $\mathcal{G}$ with the highest correlation weight, and builds the corresponding updates by adding prefix p and time t to all the update representations. In case multiple correlation groups have the same highest correlation weight, $max_{weight}(\mathcal{G}, t, p)$ selects one of them randomly. We denote $U(p, u, t)$ the set of updates reconstituted from update u with prefix p received at time t.

$$U(p, u, t) = max_{weight}(Corr(p, u), t, p)$$

Basically, BUS reconstitutes an update u by searching for the correlation group where $r(u)$ appears and that has the highest correlation weight. For each update representation in this group, BUS builds updates by adding the prefix of u and the time at which it is observed to each of them. We consider that two updates $u_1(t_1, p_1, vp_1, A_1, C_1)$ and $u_2(t_2, p_2, vp_2, A_2, C_2)$ to be identical if:

- $|t_1 - t_2| < 100$ seconds,
- $p_1 = p_2$,
- $vp_1 = vp_2$,
- $A_1 = A_2$, and
- $C_1 = C_2$

Consider a set of updates β and a subset of it α, and assume that $t(u)$ is the timestamp of an update u. The reconstitution power denoted as RP, indicates how well BUS can reconstruct β from the updates included in α. We define RP as follows:

$$RP(\beta, \alpha) = \left| \left(\bigcup_{u \in \alpha} U(p, u, t(u)) \right) \bigcap \beta \right| / |\beta|$$

$\bigcup_{u \in \alpha} U(p, u, t(u))$ is the set of updates that are reconstituted from α. Observe that this set can include updates not present in β, which we refer to as *false positives*. Incorrectly reconstituted updates occur when two updates u_1 and u_2 have the same *prefix*, *VP*, *AS path*, and *community values* but are received at time t_1 and t_2 (with $|t_1 - t_2|$ >100s) and are correlated with distinct sets of updates, resulting in u_1 and u_2 being placed in two distinct correlation groups. Consequently, reconstituting updates from u_1 might result in incorrectly reconstituting updates that appear with u_2 but with timestamp t_1, leading to false positives. Although false positives often occur, they only account for a negligible portion of the updates that could be reconstituted but are not in β (4.6%). Therefore, the false positives are ignored when computing the reconstitution power (operator $\bigcap$), which focuses only on updates in β that are correctly reconstituted (i.e., *true positives*).

BUS builds the set α of non-redundant updates by greedily adding to α the updates in $\beta \setminus \alpha$ that best improves the reconstitution power, i.e., those that maximize $RP(\beta, \alpha)$. Observe that BUS adds to α either **all** updates received by a VP or **none** at all. We show later in §5.2 that discarding redundant updates by matching them on prefix, VP, AS path, and communities leads to overfitting, and performs poorly for filtering *future* redundant updates, i.e., updates not seen in the training set. However, filtering on

prefix and VP provides the best tradeoff between discarding future redundant updates and retaining future non-redundant updates.

Reconstitution algorithm explanation with example. We use the example from Fig. 4.2 and the correlation groups built in §4.1.1 to illustrate the reconstitution algorithm. Similarly to §4.1.1, we focus on prefix p1 and omit the BGP communities.

Recall that the four events occurring in the scenario depicted in Fig. 4.2 induced eight updates $\{u_1, ..., u_8\}$. The set of initial updates β thus contains these eight updates. BUS starts by computing, for each update u in β, the set of updates reconstituted from u (denoted as U). To illustrate this process, let's consider u_1 as an example. $r(u_1)$ is included in two correlation groups, G_1 and G_3. Therefore, we have $Corr(\text{p1}, u_1) = \{G_1, G_3\}$. Since we have $w(G_1) = w(G_3)$, the function $max_{weight}(Corr(\text{p1}, u_1), T_1, \text{p1})$ selects one of them randomly; suppose it selects G_3 in our case. Finally, BUS uses the max_{weight} function to reconstitute updates by using the update representations contained in G_3 and adding p1 and T_1 to them. Thus, we have:
$U(\text{p1}, u_1, T_1) = \{(T_1, \text{p1}, \text{VP1}, 2-1-4), (T_1, \text{p1}, \text{VP2}, 6-3-1-4)\}$

BUS computes the set of reconstituted updates for all updates in β, which results in:

$U(\text{p1}, u_1, T_1) = \{(T_1, \text{p1}, \text{VP1}, 2-1-4), (T_1, \text{p1}, \text{VP2}, 6-3-1-4)\}$
$U(\text{p1}, u_2, T_1) = \{(T_1, \text{p1}, \text{VP1}, 2-1-4), (T_1, \text{p1}, \text{VP2}, 6-2-1-4)\}$
$U(\text{p1}, u_3, T_2) = \{(T_2, \text{p1}, \text{VP1}, 2-4), (T_2, \text{p1}, \text{VP2}, 6-2-4)\}$
$U(\text{p1}, u_4, T_2) = \{(T_2, \text{p1}, \text{VP1}, 2-4), (T_2, \text{p1}, \text{VP2}, 6-2-4)\}$
$U(\text{p1}, u_5, T_3) = \{(T_3, \text{p1}, \text{VP1}, 2-1-4), (T_3, \text{p1}, \text{VP2}, 6-3-1-4)\}$
$U(\text{p1}, u_6, T_3) = \{(T_3, \text{p1}, \text{VP1}, 2-1-4), (T_3, \text{p1}, \text{VP2}, 6-3-1-4)\}$
$U(\text{p1}, u_7, T_4) = \{(T_4, \text{p1}, \text{VP1}, 2-4), (T_4, \text{p1}, \text{VP2}, 6-2-4)\}$
$U(\text{p1}, u_8, T_4) = \{(T_4, \text{p1}, \text{VP1}, 2-4), (T_4, \text{p1}, \text{VP2}, 6-2-4)\}$

At the first iteration, BUS determines which update to add in α. Since it selects either all or none of the updates collected by the same VP, BUS will either add $\{u_1, u_3, u_5, u_7\}$ or $\{u_2, u_4, u_6, u_8\}$ in α. The updates collected by VP1 enable reconstituting seven out of the eight updates, while the updates collected by VP2 enable reconstituting all the updates from β since we have:

u_2 (in G_1) leads to the reconstitution of u_1 (also in G_1)
u_4 (in G_2) leads to the reconstitution of u_3 (also in G_2)
u_6 (in G_3) leads to the reconstitution of u_5 (also in G_3)
u_8 (in G_2) leads to the reconstitution of u_7 (also in G_2)

Observe that β cannot be entirely reconstituted if α contains the four updates collected by VP1. In fact, u_1 and u_5 have identical attribute values but appear correlated with

different updates throughout time (u_1 is correlated with u_2 and u_5 is correlated with u_6). Consequently, either u_2 or u_6 cannot be reconstituted. Besides, one update is inevitably incorrectly reconstituted. Either u_1 leads to reconstituting the following update:

Time T_1; VP: VP1; Prefix: pp1; AS path: 6—3—1—4

which is not in β, or u_5 leads to reconstituting the following update (which is also not it β):

Time T_3; VP: VP1; Prefix: p1; AS path: 6—2—1—4

Therefore, BUS adds four updates u_2, u_4, u_6, u_8 to α, resulting in $RP(\beta, \alpha) = 1.0$. However, in practice, it is challenging to find a set α of updates that achieve a reconstitution power of 1 while removing most of the redundant updates. We experimentally found that, based on RIS and RV data, the optimal threshold is 0.94, meaning that BUS stops when it finds a set α such that $RP(\beta, \alpha) >= 0.94$.

Reconstitution algorithm parameters. The main parameter of the reconstitution algorithm is the reconstitution power threshold at which BUS should stop adding new updates in α. Determining the optimal value for this parameter is not straightforward. On one hand, if $|\alpha|/|\beta|$ is close to one (i.e., if the proportion of retained updates is close to one), the reconstitution power is close to one (i.e., the optimum), resulting in many redundant updates being retained. On the other hand, if $|\alpha|/|\beta|$ is close to zero, the reconstitution power is low, which leads to many non-redundant updates being discarded. We experimentally find the best tradeoff between discarding as many redundant updates as possible and retaining as many non-redundant updates as possible. This tradeoff is illustrated in Fig. 4.3.

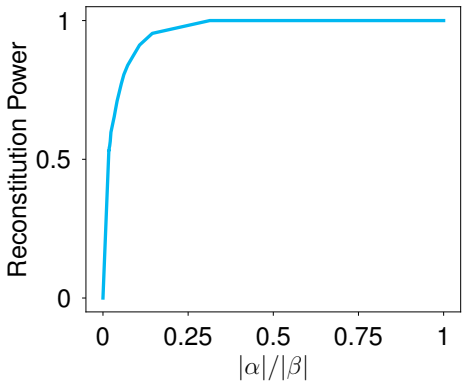


Figure 4.3: Reconstitution power of alpha as a function of the size of $\beta \setminus \alpha$.

We collect all updates from RIS and RV from Sept. 1, 2023 to Sept. 2, 2023 and add all these updates in β. We use two days of data to align with the correlation group construction time from §4.1.1. We then plot the evolution of the *reconstitution power* as a function of the proportion of retained updates (i.e., $|\alpha|/|\beta|$). As expected, the first loop of our algorithm significantly improves the reconstitution power of α, going from 0 to 0.43. As depicted in Fig. 4.3, once the reconstitution power reaches 0.94, adding new updates in α does not significantly improve it. Consequently, we choose 0.94 as the default value for the reconstitution power threshold. Note that this value can be adjusted to accommodate higher or lower processing and storage capacities. In practice, with the default parameter, we have $|\alpha|/|\beta| \approx 0.16$, meaning that the reconstitution algorithm enables to reconstitute 94% of the updates in β from $\approx$ 16% of them. BUS classifies 16% of the updates as non-redundant and the remaining 84% as redundant.

We now refine our definition of per-update redundancy to enable BUS exploiting the per-prefix redundancies and thus further reduce the set of the retained updates α.

4.1.3 Identifying redundancies across prefixes

In the introduction of this section, we described a possible strategy for sampling non-redundant updates, which consists of selecting only one update among those that share the same BGP attributes, but have different prefixes. This approach cannot be blindly applied, since it only allows for discarding non-redundant updates that appear *in the past*. However, BUS built per-prefix sets of non-redundant updates that capture consistent redundancies across time, which enables it to apply a prefix-wise redundancy reduction. Distinct prefixes can be subject to similar route updates and thus carry highly redundant information. If these redundancies are consistent enough, the set α of non-redundant updates is identical (aside from the prefix) for multiple prefixes. We then adapt BUS to exploit the redundancy between the prefixes.

BUS Prefix-wide reduction description. Intuitively, prefixes announced by the same AS are likely to carry redundant information. Consequently, for two updates u_1 with prefix p_1 seen at time t_1 and u_2 with prefix p_2 seen at time t_2 announced by the same AS, we will likely have $U(p_1, u_1, t_1) = U(p_2, u_2, t_2)$. We denote $\alpha(p)$ the set of updates in α with prefix p. BUS applies its prefix-wide redundancy reduction to a set $P = \{p_1, ..., p_n\}$ of distinct prefixes only if $\alpha(p_1) = ... = \alpha(p_n)$, with prefix excluded from the equality. We denote as $\mathcal{VP}(P)$ the set of VPs that collected at least one update in $\bigcup_{p \in P} \alpha(p)$. BUS's prefix-wide redundancy reduction consists of separating, for each prefix p in P, $\alpha(p)$ into sets of updates $\alpha(p, vp_1), ..., \alpha(p, vp_m)$ where $\alpha(p, vp)$ is the set of updates in α that have prefix p and that are collected by vp. BUS then builds a new set of non-redundant updates, denoted as α', by iteratively adding sets of updates $\alpha(p, vp)$ until α' contains at least one update collected by each VP in $\mathcal{VP}(P)$. The set

α' is built greedily by adding at each step the set $\alpha(p, vp)$ for which there is no update in α' with prefix p and no update collected by vp. In case there is no such set $\alpha(p, vp)$, BUS selects one for which there is no update in α' collected by VP vp. When there are multiple sets $\alpha(p, vp)$ that match the conditions, BUS selects one randomly. Finally, we get a new set of non-redundant updates α' where per-prefix redundancies are pruned, which results in $|\alpha'| < |\alpha|$.

BUS Prefix-wide reduction illustration (with example). We illustrate how BUS's prefix-wide redundancy reduction works using the scenario depicted in Fig. 4.2. We now assume that, similarly to the scenario in Fig. 3.1, *AS4* announces two prefixes p1 and p2. Since p1 and p2 share the same paths and experience routing changes similarly, the sets of updates with p1 and p2 added in α are the same. Therefore we have:

$$\alpha(\text{p1}) = \{(T_1, \text{p1}, \text{VP2}, 6\text{—}2\text{—}1\text{—}4), (T_2, \text{p1}, \text{VP2}, 6\text{—}2\text{—}4),$$
$$(T_3, \text{p1}, \text{VP2}, 6\text{—}3\text{—}1\text{—}4), (T_4, \text{p1}, \text{VP2}, 6\text{—}2\text{—}4)\}$$

and

$$\alpha(\text{p2}) = \{(T_1, \text{p2}, \text{VP2}, 6\text{—}2\text{—}1\text{—}4), (T_2, \text{p2}, \text{VP2}, 6\text{—}2\text{—}4),$$
$$(T_3, \text{p2}, \text{VP2}, 6\text{—}3\text{—}1\text{—}4), (T_4, \text{p2}, \text{VP2}, 6\text{—}2\text{—}4)\}$$

p1 and p2 satisfy the requirements to be grouped into the same set of redundant prefixes, since we have $\alpha(\text{p1}) = \alpha(\text{p2})$, with the prefix excluded from the equality. Consequently, we have $P = \{\text{p1}, \text{p2}\}$ and $\mathcal{VP}(P) = \{\text{VP2}\}$. We then separate the sets of updates $\alpha(p)$ into sets $\alpha(p, vp)$. Since there is only one VP in $\mathcal{VP}(P)$, we have $\alpha(\text{p1}, \text{VP2}) = \alpha(\text{p1})$ and $\alpha(\text{p2}, \text{VP2}) = \alpha(\text{p2})$.

First, BUS initializes the new set of non-redundant updates α' with an empty set. At the first step, BUS adds either $\alpha(\text{p1}, \text{VP2})$ or $\alpha(\text{p2}, \text{VP2})$, randomly selected—suppose $\alpha(\text{p1}, \text{VP2})$ here. Since there is now at least one update collected by all the VPs in $\mathcal{VP}(P)$, BUS stops adding new updates in α'. Consequently, it classifies all the updates in $\alpha(\text{p2}, \text{VP2})$ as redundant. In practice with data collected from RIS and RV, we have $|\alpha'|/|\beta| = 0.06$, meaning that BUS classifies 94% of the updates in β as redundant, and retains only the remaining 6%.

We proposed a new methodology to define the redundancy between updates, we now propose a methodology to evaluate the redundancies between VPs.

4.2 Sampling at the Vantage Point granularity

To select dissimilar VPs, a straightforward strategy is to select distant VPs. We showed at the beginning of this chapter that this strategy fails since the distance between two

VPs does not necessarily align with the redundancy of their partial views. Another strategy is to select VPs in different countries, as proposed by RIPE [115]. However, we demonstrate later in the evaluation that this strategy leads to poor performances for many common objectives, such as topology mapping. These two strategies perform poorly because they are not *data-driven*, meaning that they do not account for the real Internet routing dynamics.

In this section, we present, MVP, an algorithm that assesses the redundancy between VPs, in a pairwise fashion. Similarly to BUS, this methodology is data-driven and leverages the impact of BGP events on the partial view of each VPs. We start in §4.2.1 by describing our methodology to select an unbiased set of BGP events. Next, we leverage in §4.2.2 topological features to characterize how the VPs experience the selected BGP events. Then, we introduce in §4.2.3 the pairwise redundancy score between VPs. Finally, we explain in §4.2.4 how MVP builds a set of dissimilar VPs. We provide a summary of all the notations used in this section at the end of the chapter, see Table 5.

4.2.1 Selecting unbiased set of BGP events

MVP selects a large, unbiased set of BGP events to gauge pairwise redundancy between VPs. First, it carefully selects three types of non-global events: path changes, outages, and origin changes. MVP avoids global events since all VPs tend to see them, rendering them less discriminating. Second, it stratifies its sample of events across space and time to avoid bias.

Selecting local and partially visible BGP events. To assess redundancies between VPs, MVP focuses on BGP events that trigger topological changes:

New links: MVP focuses on events that trigger the appearance of a new AS link in the AS-level topology. For instance, a backup link used after an outage can lead to a new link appearing in the topology.

Inter-domain outages: MVP focuses on events that discard one AS link from the AS-level topology. For instance, a physical disruption on an inter-domain link can trigger such an event.

Origin changes: MVP focuses on events that change the AS at the origin of a prefix. These events can be either legitimate (e.g., multiple ASes owned by the same company doing traffic engineering) or not (e.g., MOAS hijack).

Our methodology aims at finding unique pieces of data. Since global events (i.e., seen by the majority of the VPs) do not enable the discrimination of the partial views of each VPs, MVP focuses on *partially visible BGP events*. An event is a candidate to be

added to our event set only if it is seen by less than 50% of the VPs. We tested other values ranging from 1% to 90% for this parameter and found that 50% was the one that enabled to find the most interesting events (i.e., that enable discriminating the VPs based on their redundancies with the highest consistency).

Avoiding biases across time. MVP selects events that occurred within one month. To avoid time biases, it selects only non-overlapping events (i.e., the time range of an event is not included in the time range of another event), such that two events are not tied to each other. This ensures that MVP does not overfit toward a specific period of time, and works well regardless of when the events occur.

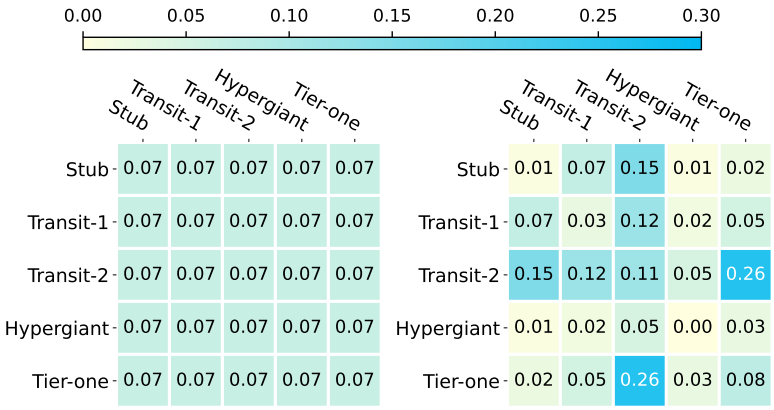
Avoiding biases across location. From a candidate set of BGP events, MVP builds the final set of events $\mathcal{E}$ by selecting 2250 non-overlapping events. We select 750 new links, 750 inter-domain outages, and 750 origin changes. We experimentally found that the pair-wise redundancy scores (described in §4.2.3) do not change when selecting more events, but are less stable when selecting fewer events. Therefore, selecting 750 of each event type provides the best tradeoff between computational expenses and redundancy stability. MVP infers the start and the end of these events by processing the data from all the VPs.

ID	Name	# of ASes	Avg.degree	Description
1	Stub	63310	3	ASes without customer
2	Transit-1	10845	27	Transit ASes with a customer cone size lower than the average
3	Transit-2	704	267	Transit ASes $\notin$ Transit-1
4	HyperGiant	15	1078	Top 15 as defined in [119]
5	Tier1	19	1817	Tier1 in the CAIDA dataset [109]

Table 2: MVP classifies all ASes in five categories, according to their characteristics.

Inspired by previous approaches to mitigate the risk of over-sampling core or stub ASes [118, 120], MVP classifies ASes into five categories, as presented in Table 2. These categories are the same as those used in [120], except for the transits that we choose to separate into two categories. The first, denoted as Transit-1, is composed of all transit ASes (i.e., ASes that have at least one customer) that have a degree lower than the average over all transit ASes, while ASes with a degree higher than the average are classified in the other category, denoted as Transit-2. This separation enables considering large and small transits separately, as they exhibit significant differences in their topological characteristics (highly vs slightly connected). If an AS satisfies the requirements for multiple categories, MVP assigns it with the highest ID.

To ensure that every scenario is represented in the set of selected events $\mathcal{E}$, MVP selects an equal number of events for every pair of AS categories. The AS pair for a new link or an AS-link outage corresponds to the ASes at both ends of the link, and for an origin change it corresponds to the previous and the new origin. Having an over-represented event scenario leads to overfitting toward that scenario, as demonstrated in [82]. Since more events are observed in the core of the Internet, having a random sampling leads to good performances when inferring redundancies at the core of the Internet, but poor performances when inferring redundancies at the edge. The results of the event selection are presented in Fig. 4.4.



(a) Balanced selection. **(b) Random selection.**

Figure 4.4: MVP selects events using a balanced selection scheme that reduces bias. The x- and y-axes are the five categories of ASes (see Table 2).

Fig. 4.4 shows the results of both a balanced (Fig. 4.4a) and random (Fig. 4.4b) selection of BGP events for each of the 15 pairs of AS categories (the matrices are symmetric) and for 2250 events selected in Sept. 2023. Unsurprisingly, the random sampling selects significantly more events in the core of the Internet (category Transit-2 — Tier-one). 69% of the selected events involve a Tier-one AS, compared to only 11% for events involving HyperGiants. However, MVP's event sampling strategy selects the same number of events in each of the 15 categories, preventing MVP from overfitting toward any specific category of event. For each of the 15 categories, our MVP selects 50 new links, 50 outages, and 50 origin changes, yielding $15 * 3 * 50 = 2250$ events ($|\mathcal{E}| = 2250$) used in the next step.

We now show how MVP uses these BGP events to quantify the observation of the VPs.

4.2.2 Characterizing how VPs observe BGP events

There exist multiple ways to evaluate the similarities between two graphs [121]. One such metric is the *graph edit distance*, which computes the minimal number of operations needed to transform the first graph it into the second. An operation is either *adding a node*, *removing a node*, *adding an edge*, or *removing an edge*. Despite efficiently embedding graph similarities, this strategy and other state-of-the-art methods for comparing graphs exhibit two main issues. First, they do not account for routing dynamics, as there are many routing changes every minute on the Internet. Second, these solutions have high complexity, meaning they do not scale well on graphs with tens of thousands of nodes. Therefore, we rely on *graph features* to evaluate the impact of the selected BGP events on the partial view of each VP.

We consider the set of VPs V that includes all VPs from RIS and RV. To compute the partial view of VP v, MVP computes the RIB of VP v at time t using its last RIB dump before t and subsequent updates until t. MVP uses this RIB to construct and maintain the directed weighted graph G_v (t) = $(N_v(t), E_v(t))$ from the AS paths of the best routes observed by v at time t, with $N_v(t)$ the set of nodes and $E_v(t) \in N_v(t) * N_v(t)$ the set of AS links. The edges are directed because two identical paths in opposite directions should not appear as redundant. Each edge in $E_v(t)$ has a weight in Z^+ which is the number of routes in the RIB that include this edge in their AS path.

MVP considers the four main BGP attributes. MVP evaluates the impact of each event on the topological features [122, 123, 124] computed on graph $G_v(t)$ for all VPs. The combination of these topological features prevents overfitting as the graphs on which they are computed embed information about the four main attributes contained within a BGP (time, prefix, AS path and, community). More concretely, the graphs $G_v(t)$ embed information about *(i)* the time as the graph is built until a given time, *(ii)* the AS path as it is used to build the AS graph, *(iii)* the prefixes, which are used to weight every edge on the graph, and *(iv)* the community values, which strongly correlate with the AS path. We confirm this correlation by downloading the first RIBs of Sept. 2023 for all VPs and analyzing the correlation between the AS path and the set of BGP communities. We find that two identical AS paths share the exact same set of BGP communities in 93% of the cases. MVP thus does not embed more information about BGP communities in $G_v(t)$ because many of them encode local traffic engineering decisions [23] that could lead to overfitting. We validate this design choice in the evaluation (§6).

MVP uses 15 diverse topological features. Topological features have been extensively used in other works to capture routing dynamics [82, 125, 126]. MVP computes topological features extracted from the literature that we listed in Table 3. These fea-

Type	Categorie	Name	Weighted	Index
Node-based	**Centrality Metrics**	Closeness centrality	✓	0
		Harmonic centrality	✓	1
	Neighborhood Richness	Average neighbor degree	✓	2
		Eccentricity	✓	3
	Topological Pattern	Number of Triangles	×	4
		Clustering	✓	5
Pair-based	**Closeness Metrics**	Jaccard	×	6
		Adamic Adar	×	7
		Preferential attachment	×	8

Table 3: Node-based and pair-based features.

tures are either *node-based* or *pair-based*. MVP computes node-based features for the two ASes involved in each event and computes the pair-based for the AS pair. MVP uses six node-based features that we classify into three categories. The first one quantifies how central and connected a node is; the second quantifies how connected are the neighboring nodes; and the third quantifies the topological patterns that include the node. We classify the three pair-based features into a single category that measures how close two nodes are based on their neighboring nodes. Five features rely on edge weights. We omit other topological features, such as eigenvector centrality, as they are redundant with the selected ones.

MVP computes the impact of each event on the features of each VP. Consider an event $e \in \mathcal{E}$ that involves two ASes e_{AS1} and e_{AS2}, starts at time e_s, and ends at time e_e. Let v be a VP $\in V$. MVP computes feature values depending on the feature type. We denote F_n (resp. F_p) as the set of node-based (resp. pair-based) features and show how MVP computes the value of these two types of features for event e and VP v.

Node-based features: Consider feature $f_i \in F_n$ and $f_i(x, G_v(t))$ its value for node x on graph $G_v(t)$, with i the feature index in Table 3. MVP computes the following 12-dimensional feature vector.

$$\begin{aligned} T_{node\_based}(v,e) = [&f_0(e_{AS1}, G_v(e_s)) - f_0(e_{AS1}, G_v(e_e)),\\ &f_0(e_{AS2}, G_v(e_s)) - f_0(e_{AS2}, G_v(e_e)),\\ &\ldots, f_5(e_{AS1}, G_v(e_s)) - f_5(e_{AS1}, G_v(e_e)),\\ &f_5(e_{AS2}, G_v(e_s)) - f_5(e_{AS2}, G_v(e_e))] \end{aligned}$$

Pair-based features: Consider feature $f_i \in F_p$ and $f_i(x_1, x_2, G_v(t))$ its value for the node pair (x_1, x_2) on the graph $G_v(t)$, with i the feature index in Table 3. MVP

computes the following 3-dimensional feature vector.

$$T_{pair\_based}(v,e) = [f_6(e_{AS1}, e_{AS2}, G_v(e_s)) - f_6(e_{AS1}, e_{AS2}, G_v(e_e)), \\ \ldots, f_8(e_{AS1}, e_{AS2}, G_v(e_s)) - f_8(e_{AS1}, e_{AS2}, G_v(e_e))]$$

The final feature vector is $T(v,e)$, a 15-dimensional vector that concatenates ($\oplus$) the node- and pair-based features.

$$T(v,e) = T_{node\_based}(v,e) \oplus T_{pair\_based}(v,e)$$

MVP now leverages these vectors for all the VPs in order to compute a *pairwise redundancy score* for each pair of VPs.

4.2.3 Computing a pairwise redundancy score

As MVP formalizes and quantifies how VPs experience BGP events, it can now leverage this information to compute a pairwise redundancy score between each pair of VPs. More precisely, for each event e in $\mathcal{E}$, MVP computes the *Euclidean distance* between each pair of VPs in an n-dimensional space, where n is the number of features used to characterize how VPs observe e (15 in our case). Intuitively, redundant VPs are likely to experience BGP events similarly, and thus to have similar feature values, resulting in them being close in the n-dimensional space. Consequently, the Euclidean distance in the topological feature space between two redundant VPs is likely to be low. MVP computes the Euclidean distance for every pair of VPs and over all selected BGP events.

We now explain in detail the three steps used by MVP to compute pairwise redundancy scores. First MVP normalizes the feature vector to uniformize the impact of each feature. Next, it computes the Euclidean distance between VPs. Finally, MVP computes the average Euclidean distances across all selected events.

Step 1: Normalize feature vectors. We selected a wide range of features that aim at capturing different aspects of the topological changes induced by routing dynamics. Consequently, all features must have the same impact on the redundancy scores. However, not all the features have the same scale. For instance, the average value for the *average neighbor degree* (f_2) is 49.7, while the average value for the *clustering* (f_5) is 0.03, meaning that the average neighbor degree has more impact when computing the Euclidean distance. Therefore, MVP relies on *data normalization* to prevent the redundancy score from overfitting toward some topological features. Multiple works focusing on BGP have already applied normalization to various features extracted from BGP dataset, especially when the objective is detecting BGP anomalies [72, 127, 128, 129]. There exist three most popular normalization strategies:

- *Min-Max scaler* [130] transforms the features to fit into a specific range while maintaining the same distribution. For each point X in the dataset, $X_{normalized} = (X - X_{min})/(X_{max} - X_{min})$ where X_{max} is the highest value in the original dataset and X_{min} is the lowest.
- *Z-score scaler* [131] assumes a Gaussian distribution of the points and transforms each point X in the dataset such that the mean of the normalized dataset is 0 and the standard deviation is 1. For each point X in the original dataset, $X_{normalized} = (X - \upsilon)/\sigma$ where υ is the mean of the original dataset and σ its standard deviation.
- *Log scaler* [132] converts all the points of the original dataset into a logarithmic scale. This normalization is useful when dealing with data points that span multiple orders of magnitude. For each point X in the original dataset, $X_{normalized} = log(X)$.

Although there exist other normalization techniques, we only presented the most popular ones. Since we cannot assume any specific distribution type for the topological feature values and we want each feature to fit into the same range, we choose to apply a Min-Max scaler. MVP normalizes the data for each event e in $\mathcal{E}$ using the matrix $\mathcal{M}(e)$ which includes the feature vectors for all VPs in V (one per line), computed on event e.

$$\mathcal{M}(e) = \begin{bmatrix} T(v_0, e) \\ \cdots \\ T(v_{|V|}, e) \end{bmatrix}$$

MVP then normalizes (operation $\triangledown$) each matrix $\mathcal{M}(e)$ column-wise using the Min-Max scaler, transforming every column such that every value is between 0 and 1. In matrix $\triangledown(\mathcal{M}(e))$, all features are in the same range, enabling a fair comparison between different features.

Step 2: Compute Euclidean distance between VPs. MVP uses the normalized matrix $\triangledown(\mathcal{M}(e))$ to compute the Euclidean distance between every pair of VPs and for each event e in $\mathcal{E}$ (operator $\diamond$). We denote $\triangledown(\mathcal{M}(e))_x$ the x-th row of the matrix $\triangledown(\mathcal{M}(e))$ and $\triangledown(\mathcal{M}(e))_{x,i}$ its value at index i (i.e., the i-th column). $\triangledown(\mathcal{M}(e))_{x,i}$ thus denote the value of feature f_i of VP x for event e. We define the Euclidean distance between the n-th VP v_n and the m-th VP v_m for event e as follows:

$$\diamond(v_n, v_m, e) = \sqrt{\sum_{i=0}^{15} (\triangledown(\mathcal{M}(e))_{n,i} - \triangledown(\mathcal{M}(e))_{m,i})^2}$$

MVP now leverages these Euclidean distances and aggregates them to compute the *pairwise redundancy scores* across all events.

Step 3: Compute the average distance over all selected events. The pairwise redundancy score $\mathcal{R}(v_n, v_m)$ between two VPs v_n and v_m relates to the normalized average Euclidean distance between them across the 2250 events in $\mathcal{E}$, computed as:

$$\mathcal{R}(v_n, v_m) = 1 - \coprod((\sum_{e \in \mathcal{E}} \diamond(v_n, v_m, e)) * \frac{1}{|\mathcal{E}|})$$

The operator $\coprod$ applies a Min-Max scaler to the computed value, such that all the redundancy scores are between 0 and 1. The max and the min in $\coprod$ corresponds to the highest and the lowest possible value for $\mathcal{R}(v_n, v_m)$ for every possible $(v_n, v_m) \in V * V$. A redundancy score of 0 does not mean that the pair of VPs is not redundant at all, but that this pair exhibits the lowest rate of redundancy among all possible pairs of VPs. Conversely, a value of 1 means that the two VPs are the most redundant VPs.

We defined the pairwise redundancy scores between two VPs, we show now how MVP leverages these redundancy scores to build the set of least redundant VPs.

4.2.4 Computing a set of dissimilar VPs

MVP builds a set of the least redundant VPs by leveraging both the *redundancies* between VPs and the *volume* of collected data. MVP considers the volume of data generated by every VP to prioritize those that provide richer information within fewer updates. MVP initializes the set of least redundant VPs S by adding the most common VP from V, i.e., the VP that minimizes its average distance to other VPs. More precisely, MVP initializes S with the VP having the lowest average distance $AD(v)$, defined as follows:

$$AD(v) = \frac{1}{|V| - 1} * (\sum_{v' \in V \setminus v} \mathcal{R}(v, v'))$$

This design choice enables the redundant part of the BGP data (e.g., c2p links that are observed by most of the VPs) to be visible with the first selected VP. Thus, it is easier to add VPs that see less but unique information about the Internet routing ecosystem. Then, MVP greedily adds the VP that balances maximal Euclidean distance to VPs already selected (i.e., in S) and minimal additional volume that the VP brings. More formally, at each iteration, MVP builds a candidate set of VPs $\mathcal{K}$ that contains the non-selected VPs exhibiting the lowest maximum redundancy score. The maximum redundancy score, denoted as P measures the maximum redundancy between a VP v

and the set of VPs S and is defined as follow:

$$P(S, v) = max(\mathcal{R}(v, v'), \forall v' \in V)$$

MVP adds to $\mathcal{K}$ the $\gamma = 10\%$ of the non-selected VPs that exhibit the lowest maximum redundancy score. It then adds to set S the VP in candidate set $\mathcal{K}$ that collects the lowest volume of data compared to the other VPs in $\mathcal{K}$. This enables MVP to select VPs that achieve a good balance between volume of collected data, and unique information added. We estimate the volume of data collected by the VPs by counting the number of BGP updates that they received during a 365-day period. We reduce the computational expense of this analysis by selecting (randomly) only one hour of data every day. The γ parameter allows tuning redundancy and volume knobs: a low γ prioritizes low redundancy while a higher γ prioritizes low resulting data volume. We found that γ = 10% performs well in practical scenarios (we tested a range from 1% to 50%).

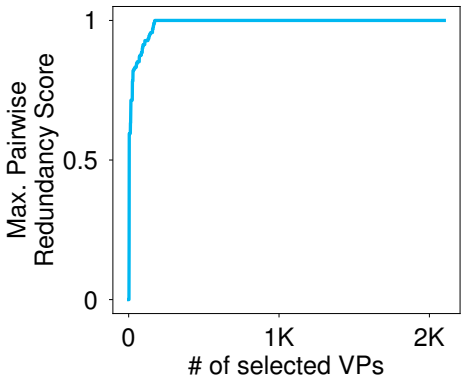


Figure 4.5: Evolution of the minimal "maximal redundancy" between non-selected VPs and selected VPs.

MVP stops adding new VPs in S when the minimal maximal redundancy score reach 1, meaning that it is impossible to add a non-selected VP that is not heavily redundant with at least one VP in S. We define the minimal maximal redundancy score MMR as follow:

$$MMR = min(P(S, v), \forall v \in V \setminus S)$$

In practice with RIS and RV data, MMR reaches 1 when $|S| = 178$.

In this chapter, we presented two algorithms, BUS and MVP, that evaluate the redundancy in BGP data in a general fashion. BUS evaluates the redundancies at the granularity of the BGP update, while MVP evaluates the redundancies at the granularity of the VP. We now present GILL, a system that filter the redundant BGP data using an overshoot-and-discard collection strategy.

Notation	Name	Description
$u(t, p, vp, A, C)$	BGP update	BGP update represented by the time t at which it is received, the prefix p it carries, the VP vp that collects it, the associated AS path A and the associated communities C.
$r(u)$	BGP update representation	Representation of a BGP update u within a correlation group. The correlation groups are built per prefix across time, thus $r(u) = (vp, A, C)$.
$w(G)$	Correlation group weight	Function that associate a correlation group G to its correlation group weight. The weight is given by the number of times the same set of updates appear correlated in time.
β	Initial set of updates	Original set of BGP updates, prior any redundancy reduction.
α	Set of retained updates	Set of updates after redundancy reduction, also refered as set of non-redundant updates.
$Corr(p, u)$	List of correlation groups	Lists the correlation groups for prefix p and that include thre representation $r(u)$ of update u.
$max_{weight}(\mathcal{G}, t, p)$	Selected correlation group	Function taking as input a set $\mathcal{G}$ of correlation groups and builds updates by adding time t and prefix p to the update representations in the group in $\mathcal{G}$ with the highest weight.
$U(p, u, t)$	Reconstituted updates	Set pf reconstituted updates from update u oberved at time t with prefix p.
$RP(\beta, \alpha)$	Reconstitution Power	Computes the reconstitution power of a subset α of β. The reconstitution power ignores incorrectly reconstituted updates.
α'	Reduced subset of updates	Subset of β where the redundancies between prefixes have been pruned.

Table 4: Summary of all notations used to define the redundancies between updates. For more formal details refer to §4.1.

Notation	Name	Description
$\mathcal{E}$	Set of BGP events	Set of selected BGP events. The events are selected using a balanced sampling to prevent overfitting.
$G_v(t)$	VP partial view	Graph computed using the RIB at time t from VP v. The AS topology is build using the collected AS paths.
$f_i(x, G_v(t))$	Topological feature value	Corresponds to the value of the feature at index i on Table 3, computed of graph $G_v(t)$ using AS x as a source.
$T(v, e)$	Feature vector	Vector obtained by computing the impact of event e on the partial view of VP vp. The values in the vector are stored accoring to the index of the features.
$\mathcal{M}(e)$	Event matrix	Matrix computed on event e, where each line is the feature vector of one VP.
$\triangledown$	Column-wise normalization	Operation that normalize an event matrix column-wise, using a Min-Max scaler. The objective is to put all the features on the same scale, enabling a fair comparison.
$\diamond(v_n, v_m, e)$	Euclidean distance	Operation that compute the euclidean distance between the feature vectors of VPs v_n and v_m for event e. Euclidean distances are computed in a 15-dimensional space.
$\mathcal{R}(v_n, v_m)$	Pairwise redundancy score	Computes the pairwise redundancy score between VPs v_n and v_m.
$AD(v)$	Average Distance	Computes the average redundancy score between VP v and all other VPs.
$P(S, v)$	Maximum redundancy score	Returns the maximal value among all redundancy scores between VP v and all VPs in S.
MMR	Minimal Maximal redundancy score	Returns the minimal value among all maximal redundancy scores of the VPs that are not selected in the set S of least redundant VPs.

Table 5: Summary of all notations used to define the redundancies between VP. For more formal details refer to §4.2.

5

GILL: identifying redundancy in BGP data

Current data collection and monitoring platforms leverage a limited number of VPs. Excluding commercial platforms where the number of VPs is confidential ([97, 100]), data collection systems peer with ≈ 2k VPs, yielding a VP coverage of less than 2%. However, as demonstrated earlier in §2, a low VP coverage leads to significant visibility gaps for many objectives. Our simulations suggested that reducing this gap requires an increase in VP coverage of at least an order of magnitude. Unfortunately, the prohibitive volume of collected data hinders the deployment of new VPs. The reality is that no existing system today can accommodate a substantial increase in VP coverage. Indeed, current systems are constrained by the inherent nature of BGP data. The main contribution of this thesis is GILL, a new BGP data collection system that can collect data from an order of magnitude more VPs compared to the current systems, while effectively managing data volume constraints.

Current BGP data collection platforms collect **all** the data from a **limited** number of VPs. GILL shifts this data collection paradigm by using an *overshoot-and-discard* strategy, which involves collecting data from as many VPs as possible and then discarding the redundant data prior to storing it. GILL relies on the redundancy definitions presented in §4, therefore enabling most of the measurement studies to perform effectively. In this chapter, we describe the design and implementation of GILL.

We start in §5.1 by describing the overshoot-and-discard approach, explaining why it can be applied to BGP data, and outlining its benefits. In §5.2, we explain how GILL builds and maintains filters that enable discarding redundant BGP data while retaining the most useful information. Finally, we present in §5.3 the implementation of GILL, which prototype currently works with original peers as well as data from RIS and RV. This thesis focuses on the operational challenge induced by a drastic increase in VP coverage. We provide later some possible ideas on how to incentivize network operators to peer with GILL.

5.1 An overshoot-and-discard approach for BGP data

The overshoot-and-discard approach for BGP data consists of two steps: *(i)* collecting BGP routing data from as many VPs as possible and *(ii)* discard the redundant portion before any further processing. As this new strategy implies a shift in the BGP data collection paradigm, we acknowledge that some concerns may arise from the scientific community. However overshoot-and-discard collection strategies have already been considered and adopted in other research areas.

High-Throughput Screening: High-Throughput Screening (HTS) [133] facilitates faster analysis of biological components. This approach has gained popularity and become the standard method for drug discovery in the pharmaceutical industry [134]. This technique uses dedicated hardware and algorithms to process more than 100k compounds daily by focusing only on the most meaningful portion of the data and discarding the remaining data before further analysis.

Astronomic analysis: Astronomic sensors collect high volumes of data, which researchers often cannot afford to process. To manage these issues, they reduce the volume of data to process by eliminating the high proportion of noise in the data. Researchers have developed algorithms that focus only on the most meaningful piece of the data. These strategies can rely on signal processing methods such as wavelets [135] or Fourier Transforms [136] to separate noise from the valuable piece of the collected data.

Remote Sensing: Remotely sensed data consists of images taken by aerial sensors, collected and stored in databases. This data can be used for various objectives; mapping road networks, analyzing forests, studying oceans, and so on. Because of their high volume, users often cannot afford to process all the images collected by the sensors and thus rely on algorithms that extract the most relevant images. For instance, specific algorithms enable focusing on the road images, enhancing performances when mapping road networks [137].

Large Hadron Collider: The Large Hadron Collider (LHC) is the world's largest particle collider, built by the European Organization for Nuclear Research. The collider

generates millions of collisions just to observe a few interesting ones, e.g., the Higgs boson [138]. To focus only on the most relevant collisions, researchers rely on fast online algorithms and custom hardware to discard 99.994% of the likely less interesting collisions [139].

As illustrated by these examples, an overshoot-and-discard approach can be applied to datasets where most of the signals are either redundant or considered noise. Since BGP fits in this category (as presented in §3), we illustrate how this new data collection strategy can be applied in the context of BGP routing data and explore its benefits. We then introduce GILL, a BGP data collection system that implements the overshoot-and-discard strategy.

5.1.1 Overshooting and discarding BGP data

The data collected by the BGP collection platforms such as RIS and RV exhibits a high level of redundancy, making it an ideal candidate for an overshoot-and-discard collection strategy. We illustrate using an example how this data collection strategy can be beneficial in the context of BGP.

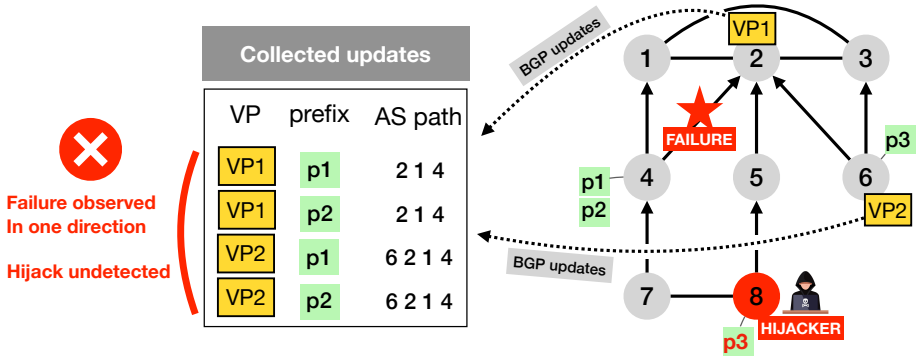


Figure 5.1: Collection of updates with the current approach. The collected updates do not enable to observe the failure in both direction nor to detect the hijack intended by *AS8*.

Overshoot-and-discard strategy with an example. Using the scenario depicted in Fig. 5.1, we illustrate the current approach for collecting BGP data and its limitations. We consider an eight-AS topology where the routing policies are configured according to the Gao-Rexford policies. *AS4* announces two prefixes, p1 and p2, while *AS6* announces p3. We consider a deployment of two VPs, VP1 located in *AS2* and VP2 located in *AS6*. Both these VPs export their best routes to a route collector. We consider two distinct events occurring roughly at the same time: *(i)* an outage on link 2—4 and

(ii) a MOAS hijack on p3, intended by *AS8*. These two events trigger the propagation of four BGP updates:

Update seen by VP1, with prefix p1 and AS path 2—1—4,

Update seen by VP1, with prefix p2 and AS path 2—1—4,

Update seen by VP2, with prefix p1 and AS path 6—2—1—4, and

Update seen by VP2, with prefix p2 and AS path 6—2—1—4

There are three main issues with the four collected updates. *(i)*, there is a high level of redundancy in the set of updates collected by the two VPs. Indeed, the AS paths of the two routes observed by VP1 are included in the AS paths observed by VP2. Additionally, the BGP attributes announced with p1 and p2 are the same. *(ii)*, the collected updates enable to observe the failure in only one direction, as all the collected paths circumvent the failure going from *AS4* to *AS2*, but none of them circumvent the failure going from *AS2* to *AS4*. Observing the failure in only one direction prevents failure localization algorithms [111] from producing accurate inferences. *(iii)*, none of the VPs observe the hijacked route, as it does not propagate over *AS5* and *AS7*, due to routing policies and best path selection. Therefore, the collected updates do not enable hijack detection systems such as ARTEMIS [69] to detect the hijack. Redundancy in the set of collected updates increases the volume of data, while not providing any additional relevant information about the routing system.

We now illustrate the benefits of the overshoot-and-discard strategy in the context of BGP data. We consider the same AS topology, routing policies configuration, prefixes announced, and BGP events as used in the scenario depicted in Fig. 5.1. The first step of this data collection strategy is to *overshoot* the BGP data, meaning that we collect data from as many VPs as possible. We thus assume a deployment of four VPs, VP1 in *AS2*, VP2 in *AS6*, VP3 in *AS4*, and VP4 in *AS5*. The scenario is depicted in Fig. 5.2. The two additional VPs collect four new BGP updates:

Update seen by VP3, with prefix p3 and AS path 4—1—2—6,

Update seen by VP4, with prefix p1 and AS path 5—2—1—4,

Update seen by VP4, with prefix p2 and AS path 5—2—1—4, and

Update seen by VP4, with prefix p3 and AS path 5—7

With eight collected updates, we can now observe the failure in both directions (thanks to the update collected by VP3) and detect the hijack (thanks to the update collected by VP4 for p3). While these new updates provide valuable information, they also increase the volume of collected data. Therefore, we now apply the *discard* step by

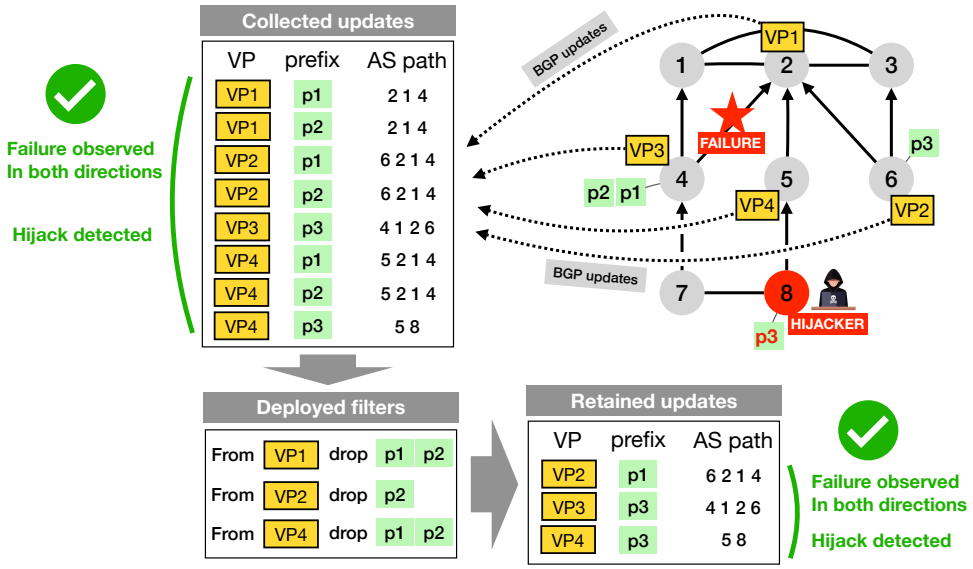


Figure 5.2: Collection of updates with the overshoot and discard approach. The updates collected by the additional VPs enable to observe the failure in both directions and to detect the hijack. The retained routes after applying the filters also enable to achieve both objectives.

deploying filters on the route collectors that drop the redundant updates. We configure the following filters:

From VP1, drop all updates for p1 and p2.

From VP2, drop all updates for p2.

From VP4, drop all updates for p1 and p2.

With these filters deployed on the route collectors, only three updates are retained. Because the deployed filters aim at discarding only redundant updates, the retained updates still contain all the valuable information. Despite collecting fewer updates compared to the actual BGP data collection strategy, we can still observe the failure in both directions and detect the hijack.

Given the high redundancy in data collected by BGP collection platforms, a significant portion of the data can be discarded before any further processing or storage process, while minimizing information loss. We now introduce the main contribution of this thesis: GILL, a system that implements this overshoot-and-discard strategy to collect only the most valuable BGP data. We present the key design choices of GILL, which we will evaluate in the next chapter (§6).

5.1.2 GILL's design choices

In the example depicted in Fig. 5.2, we purposively placed the two additional VPs and optimized filters to detect the two routing events and discard updates with similar attributes. For example, the four updates that VP1 and VP2 collect for p1 and p2 have similar AS paths; GILL retains only one of them. In practice, deciding which updates to discard and which updates to retain is challenging. There is no ground truth about which routing events will appear where, how they will propagate, and what users want to do with the data.

The problem of strategically placing new VPs has been extensively studied over the last two decades. These different works suggest deploying a few but strategically positioned VPs, which corresponds to RIS's current VP deployment strategy [115]. For example, Roughan et al. [24] used an Expectation Maximization algorithm to estimate the number of VPs that should be deployed. They found that 700 VPs strategically placed may be capable of observing 99% of the AS links. Gregori et al. [8] used a new metric, the c2p distance, to identify the ASes likely to provide the most unique partial view over the BGP routes. Cittadini et al. [7] demonstrated the marginal utility of adding new VPs in the core of the Internet. Finally, Zhang et al. [6] highlighted the impact of the VP deployment on three different measurement studies: AS topology mapping, Hijack detection, and AS relationship inferences.

In this thesis, we do not address the challenge of placing new VPs, as it has been extensively studied in previous research. Moreover, the overshoot-and-discard collection strategy does not require a few strategically selected VPs, but a significant increase in the VP coverage. GILL then implements algorithms to reduce the volume of collected data by discarding the redundant data. Thus, for any newly deployed VP, GILL's algorithms enable our system to retain only the unique piece of information collected by this VP.

GILL leverages a key property of the redundancy in BGP data highlighted in §4.1.1: *"A significant proportion of the events that trigger BGP updates are recurrent, and correlations between past updates are likely to reappear in the future."*. GILL identifies redundancies in *past* BGP data and builds filters that enable filtering *future* redundant updates. While discarding data inevitably results in information loss, GILL's filters aim at retaining the most useful updates (regardless of the user's objective) and discarding redundant updates, thereby mitigating the impact of sampling data on the numerous studies conducted using BGP data. To achieve this, GILL samples past updates based on their redundancy and builds filters that enable discriminating the non-redundant updates

from the redundant ones. GILL uses two key ingredients to minimize information loss and enable effectively conducting many objectives.

Key ingredient #1: Support for a flexible definition of redundancy. We demonstrated that naive or specific definitions of redundancy inevitably fail for one or more objectives GILL uses the update redundancy definition provided in §4.1 to determine which updates should be kept and which can be discarded. It uses the same parameters as BUS to classify updates as redundant or non-redundant (i.e., updates in α'). As illustrated in Fig. 5.2, filtering redundant updates significantly reduces the volume of collected data while minimizing information loss. However, even with the retained updates, some measurement studies cannot be conducted accurately. For instance, *AS4*'s operator wants to determine whether all the ASes in the routing system handle all its prefixes in the same way, this objective is unachievable with the collected updates. In fact, because of the deployed filters, no VP in the topology retains updates for both p1 and p2. To ensure fairness for all possible objectives, GILL retains all the data from a subset of the VPs, which we call *anchor VPs*. This means that although these anchor VPs collect redundant updates, GILL will not configure any filters on them.

We demonstrated in §3.2 that redundancy can also occur between VPs. Therefore, GILL relies on a per-VP redundancy definition to select only the least redundant VPs, striking the best balance between additional data volume (as no filters are configured for these VPs) and redundancy elimination.

Key ingredient #2: Retain all updates from a few valuable VPs. Some studies require data for all prefixes even if redundant, which is the case for the previous example where *AS4* wants to know how the rest of the Internet handles its prefixes. Therefore we modify the example depicted in Fig. 5.2, where GILL now retains *all* updates from VP2, by removing all filters configured on VP2. With these new filters, GILL collects the same number of updates as the current BGP data collection strategy (four), but the collected updates enable more useful inferences. Specifically, the three objectives are achievable: *(i)* observing the failure in both directions, *(ii)* detecting the hijack, and *(iii)* determining if other ASes in the routing system handle p1 and p2 similarly. In practice, it is challenging to know from which VPs we should retain all updates, as we show in §4.2 that selecting VPs naively often results in selecting redundant VPs. GILL relies on the per-VP redundancy definition that we presented in §4.2, using the same parameters as MVP to find which VPs should retain all collected updates.

We explained how GILL selects the updates to retain. We now show how it leverages these updates to build filters that enable retaining only non-redundant updates while discarding redundant updates. Additionally, we show how the filters are updated over time to accommodate changes in the routing dynamics.

5.2 Filtering redundant updates

Filtering is a process available in most of the BGP implementations, used for various purposes, such as filtering attacks [140], controlling IP spoofing packets [141], implementing routing policies [142], or filtering BGP updates that carry a prefix larger than /25 (which is not allowed in BGP). There also exist tools that generate filters to discard suspicious announcements, i.e., those that do not conform to the data stored in the IRR [143, 144]. Filter implementations often allow the deployment of filters that match the prefix, the sender IP, or various BGP attributes. For instance, filters can match a subset of the AS path to prevent the propagation of some links on the Internet, which can be useful to mitigate Forged-Origin hijacks. However, it is challenging to determine which attributes GILL's filter should match the BGP updates on to reduce the redundancy. On one hand, filtering on all attributes may overfit to filter future redundant updates (i.e., that have not been seen in the training data). On the other hand, filtering on a few attributes may retain redundant updates.

We start by describing in §5.2.1 how GILL generates filters that balance retaining future non-redundant updates and filtering future redundant updates. We then present in §5.2.2 the stability of the generated filters and the frequency at which GILL should update them.

5.2.1 GILL's filter generation

GILL builds filters that aim at discarding redundant updates that are not collected by an anchor VP while retaining all others. GILL employs an *accept everything* default filtering policy. This approach ensures to retain updates containing new pieces of information, i.e., never seen in the past Additionally, updates collected by newly deployed VPs are retained until the filters are recomputed. There are three different types of filters, listed below in decreasing order of priority:

1. Retain all updates from anchor VPs
2. Discard updates that match the update filters
3. Accept everything

The first filter ensures that all data from the anchor VPs is retained, allowing any objectives that require complete data, even if redundant, to be achievable. The second filter discards all redundant updates, while the last enables to retain all updates containing new unique pieces of information. In this section, we focus on the second

filters (called *update filters*), discussing how they are generated and determining their optimal granularity.

Filter generation process. GILL leverages the updates classified as redundant by BUS (i.e., $\beta \setminus \alpha'$) to build filters that discard redundant updates. Essentially, GILL builds filters that discard any combination of attributes that appear in at least one update classified as redundant, depending on the filter granularity.

Me now formalize the filter generation process. We denote as $u(t, vp, p, A, C)$ the update u received at time t, from VP vp with prefix p, AS path A, and communities C. We denote as $attr(u, g)$ the function that associates an update u to the set of attributes considered by granularity g. As an example, using the scenario of Fig. 4.2, $attr(u_1, (vp, p)) = (\boxed{\text{VP1}}, \text{p1})$. Note that the timestamp is excluded from the $attr$ function since matching the timestamp might result in future redundant updates being retained. We denote $BF(attr)$ the function that takes as input a set of attributes and builds the corresponding filters. BF builds a filter that matches all attributes considered in $attr$. Therefore, for a given granularity g, GILL builds the set of filters $\mathcal{F}(g)$ as follow:

$$\mathcal{F}(g) = \bigcup_{u \in \beta \setminus \alpha'} BF(attr(u, g))$$

where operator $\bigcup$ performs the union of the generated filters and removes any duplicates. Duplicate filters occur when $attr(u_1, g) = attr(u_2, g)$ for two distinct updates u_1 and u_2. For example in the scenario depicted in Fig. 4.2, $attr(u_1, (vp, p)) = attr(u_3, (vp, p))$.

To illustrate how GILL builds the filters, we use the example depicted in Fig. 5.2. Similarly to §4.1, we simplify the example by not considering the community values, as it does not fundamentally change the filter generation process. We consider the granularity $g = (vp, p)$, meaning that the generated filters match the receiving VP and the prefix, to align with the example of Fig. 5.2. As illustrated, eight updates are collected:

u_1: observed by VP VP1 for prefix p1 with AS path 2—1—4
u_2: observed by VP VP1 for prefix p2 with AS path 2—1—4
u_3: observed by VP VP2 for prefix p1 with AS path 6—2—1—4
u_4: observed by VP VP2 for prefix p2 with AS path 6—2—1—4
u_5: observed by VP VP3 for prefix p3 with AS path 4—1—2—6
u_6: observed by VP VP4 for prefix p1 with AS path 5—2—1—4
u_7: observed by VP VP4 for prefix p2 with AS path 5—2—1—4

u_8: observed by VP VP4 for prefix p3 with AS path 5—8

We suppose that α' is composed of $\{u_3, u_5, u_8\}$ since the remaining updates are highly redundant with updates in α'. Therefore, we build for every update in $\beta \setminus \alpha'$ their attributes with granularity $g = (vp, p)$:

$attr(u_1, g) = (\text{VP1}, \text{p1})$
$attr(u_2, g) = (\text{VP1}, \text{p2})$
$attr(u_4, g) = (\text{VP2}, \text{p2})$
$attr(u_6, g) = (\text{VP4}, \text{p1})$
$attr(u_7, g) = (\text{VP4}, \text{p2})$

Finally, when computing $\mathcal{F}(g)$, we have:

- From VP1, discard p1,
- From VP1, discard p2,
- From VP2, discard p2,
- From VP4, discard p1, and
- From VP4, discard p2

Which enables collecting only non-redundant updates. However, in practice, it is not straightforward to select the granularity at which GILL should build the filters, as it significantly impacts the redundancy of the collected updates.

Filter generation granularity. Due to the "accept everything" default policy, the granularity of the update filters significantly impacts the quality of retained updates. Filtering on fewer attributes (e.g., only on the collecting VP) results in many non-redundant updates being discarded, but enables to effectively filter future redundant updates. Conversely, filters matching all attributes (i.e., collecting VP, prefix, AS path, and communities) enable retaining most of the future non-redundant updates but perform poorly in discarding future redundant updates. Since these updates are likely to appear with new attributes not seen in the training set, GILL cannot build filters that match these new attributes.

In practice (i.e., with RIS and RV data), updates classified as redundant by BUS at time t_1 are often similar to updates classified as redundant at time t_2 with $t_1 < t_2$. GILL generates coarse-grained filters that match the prefix and the receiving VP. Such filters match on an entire space of similar (and often redundant) updates which are either all discarded or all retained. We experimentally find that this granularity provides the best tradeoff between retaining non-redundant updates and filtering future redundant updates. We compared three versions of GILL, GILL *-pfx*, GILL *-pfx-asp*, and GILL

-pfx-asp-com. All these three versions filter based on the receiving VP, as the filters are deployed directly on the VP.

GILL *-pfx:* This version of GILL builds filters that match the receiving VP and the prefix (i.e., $g = (vp, p)$). With this granularity, an update received by a given VP is filtered only if its prefix matches one of the filters deployed on this VP. This represents the coarser granularity for the generated filters, as all the updates for a prefix not filtered are retained, regardless of the other attributes.

GILL *-pfx-asp:* This version of GILL builds filters that match the receiving VP, the prefix, and the AS path (i.e., $g = (vp, p, asp)$). All these three attributes must match exactly, meaning that an update is discarded only if all three attributes are identical to one of the generated filters. This version of the filters has an intermediate granularity since it allows different community values.

GILL *-pfx-asp-com:* This version of GILL builds filters that match the receiving VP, the prefix, the AS path, and the community values (i.e., $g = (vp, p, asp, comm)$). Similarly to GILL *-pfx-asp*, an update is discarded only if all four evaluated attributes match exactly one of the deployed filters. This version of the filters is the most fine-grained since it requires an exact match on all main attributes of a BGP update.

We utilize a typical training-testing pipeline to evaluate both the proportion of redundant updates that the generated filters effectively discard and the proportion of non-redundant updates that the filters effectively retain.

Proportion of redundant updates discarded: We start this experiment by computing the set α' of non-redundant updates computed by BUS on RIS and RV data collected from December 1 2023 to December 2, 2023. We then separate the set of redundant updates $\beta \setminus \alpha'$ into two different sets $r1$ and $r2$. $r1$ corresponds to the training set, i.e., the set of updates that GILL will use to build the filters, while $r2$ corresponds to *future updates* and will be used to evaluate the ability of the generated filters to discard redundant future updates. For this experiment, we use $|r1| = 0.8 * |\beta \setminus \alpha'|$ and $|r2| = 0.2 * |\beta \setminus \alpha'|$. We tested with other values and obtained similar results. Next, we compute, for each different filter granularity, the proportion of updates in $r2$ that match at least one of the filters built using the updates in $r1$. We perform this experiment 10 times with different time periods to compute $\beta \setminus \alpha'$, in order to avoid biases induced by update sampling.

In Fig. 5.3, we can see, very unsurprisingly, that the finer-grained the filter generation, the fewer future redundant updates in r_2 filtered. In the median case, for GILL *-pfx* (i.e., $g = (vp, p)$), 87% of the updates in $r2$ are matched by at least one filter. This number decreases with a finer granularity, 43% for GILL *-pfx-asp*, and 0% for GILL *-pfx-asp-comm*. The latter number indicates that none of the filters match any future

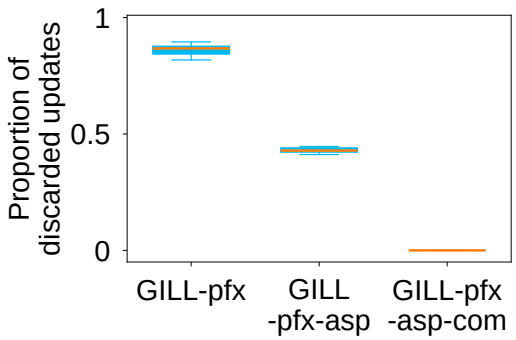


Figure 5.3: Proportion of redundant updates that are discarded by at least one of the generated filters. We present the result for gradually finer-grained filters.

redundant updates. Since these filters match all possible criteria and future updates, by definition, differ from the past updates by at least one attribute, there is no surprise that none of the updates in $r2$ are matched by any fine-grained filter. Therefore, we observe that coarse-grain filters have significantly better performances compared to other granularities. We now conduct the same experiment to determine, for each granularity, the proportion of non-redundant updates that are matched by at least one generated filter and thus discarded.

Proportion of non-redundant updates discarded: When selecting which update to add in α', GILL either adds all or none of the updates collected by a VP for a given prefix. This design choice is essential to ensure proper discrimination between redundant and non-redundant updates, especially for fine and intermediate granularities. In fact, since the updates are added in α' per prefix, if one update for prefix p collected by VP vp is added in α' while another is added in $\beta \setminus \alpha'$, it creates ambiguity in handling future updates with prefix p and VP vp, especially for fine-grained filters. As a result, no update in α' can be matched by any of the filters built using updates in $\beta \setminus \alpha'$, as at least one attribute differs for any possible granularity, by construction. Therefore, the proportion of retained future non-redundant updates is 1, for every possible granularity.

According to these results, we parameter GILL to generate filters that match the receiving VP and the prefix, as this granularity exhibits the best performance in filtering future redundant updates and all the filter granularities perform similarly when retaining future non-redundant updates. Another important reason for this design choice is to maintain the accuracy of the Routing Information Base (RIB) entries. When filtering the updates based not only on these two criteria but also on other BGP attributes, an entry for a given prefix p at a time t in the RIB might be outdated because the latest update for this prefix may have been filtered, but not the previous one.

Let's use the example of Fig. 5.2 to illustrate this. Suppose we have a fine-grained filter on VP2 for p1 that matches on AS path 6—2—4. The results depicted in the figure remain the same, and we will collect three updates. Suppose now the link 2—4 is restored. VP2 will receive a new update for p1, with AS path 6—2—4. However this update will be filtered and thus the last entry in VP2's RIB for p1 will not reflect the actual routing state for p1.

For these two main reasons, we choose to have GILL = GILL *-pfx* to generate the coarse-grained filters that match the receiving VP and the prefix. This approach ensures better performance in filtering redundant updates and maintains the consistency of the RIBs.

5.2.2 Keeping filters up-to-date

The routing dynamics evolve over time, altering the correlations between updates triggered by BGP events. For instance, two ASes renegotiating their economic relationship are likely to change their routing policies, resulting in BGP updates being announced with new AS paths. Additionally, new ASes operating in the Internet routing ecosystem may also trigger new BGP updates, as these ASes are likely to announce new prefixes, with a new path never seen before. The deployment of a new VP also brings new unique pieces of information along with redundant data. Due to GILL's accept everything default filtering policy, new pieces of redundant information that regularly appear on the Internet may generate a significant amount of new redundant updates that are not matched by any of the GILL's generated filters, resulting in a high volume of redundant data being retained. Therefore, GILL needs to regularly update the generated filters, as well as the selected anchor VPs to address these evolving dynamics.

In this section, we experimentally determine the optimal frequency at which GILL should recompute the generated filters. Two types of filters among those deployed by GILL requires regular updates. *(i)* The generated update filters need to be recomputed to accommodate changes in Internet routing dynamics. *(ii)*, the set of anchor VPs must be regularly updated due to the constant evolution of the Internet routing ecosystem. We infer how often GILL needs to recompute its filters, from experimental analysis.

Updating the generated update filters. We evaluate how accurate GILL's redundant update inference remains over time. Specifically, we compute the proportion of filtered updates, according to the time period between the filter generation and the filtering process. We build GILL's filters using data from RIS and RV, ranging from September 1, 2023 to September 2, 2023. We measure their ability to discard redundant updates in a testing set of updates collected d days after the filter generation, with d ranging from 1 to 128. The testing sets of updates are collected over two-hour periods and we present

the results for 30 different periods to avoid biases induced by time sampling. We show the results of this experiment in Fig. 5.4.

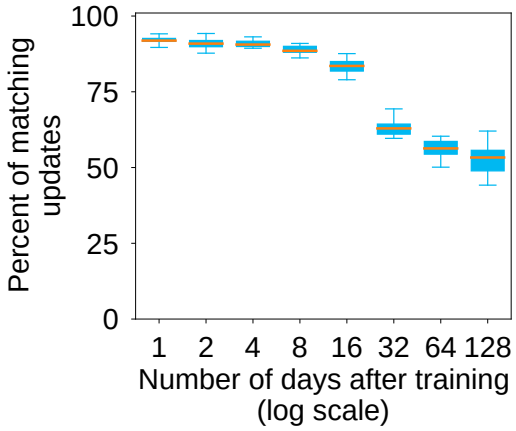


Figure 5.4: Ability of the generated filters to discard redundant updates over time. The proportion of discarded updates starts to significantly decrease after 16 days.

Unsurprisingly, the higher the value of d, the lower the proportion of matched (i.e., discarded) redundant updates. Because the routing dynamics evolve over time, the proportion of future redundant updates that none of the filters match increases, resulting in a high volume of redundant data being retained. One day after the filter generation, the proportion of discarded updates is, in the median case, 0.92 which aligns with the practical result found in §4.1.3, where we had $|\alpha'|/|\beta| \approx 0.06$. This number remains stable until $d = 16$, where the proportion of discarded updates is 84%. Beyond 16 days, this number significantly drops, reaching 64% after 32 days in the median case. Therefore, we calibrate GILL to recompute the update filters every 16 days, as this appears to be the threshold after which the proportion of discarded redundant updates significantly drops.

We now experimentally show how often GILL should update its list of least redundant VPs, known as the anchor VPs.

Updating the anchor VPs. The evolution of routing dynamics and the deployment of new VPs impact the quality of GILL's anchor VPs selection. We evaluate how often GILL should update the selected anchor VPs. Directly comparing the similarities between two sets of VPs may introduce bias in the results for two main reasons. First, since the anchor VPs are selected in a greedy fashion, a slight change in the redundancy scores can significantly alter the ordering of the VPs, even though the new order is still relevant and different least redundant VPs are selected. Second, newly deployed

VPs can alter the ordering of the selected anchor VPs, as a newly deployed VP may be included in the set, shifting the order of the remaining VPs. Therefore, we prefer to compare the evolution of the pairwise redundancy scores, as this information is indicative of the redundancies between the VPs, and their evolutions.

We compute the pairwise redundancy scores on events collected in September 2023 using MVP, and compare their evolution with the redundancy scores computed m months before. Since all pairwise redundancy scores are comprised between 0 and 1, we define the evolution of the pairwise redundancy score as the difference between the value on September 2023 and the value computed m months before. To avoid bias induced by newly deployed VPs, we only consider pairs of VPs that existed 66 months ago, the largest possible value for m. We present the result of this analysis for m ranging from 6 to 66 (=5.5 years) in Fig. 5.5.

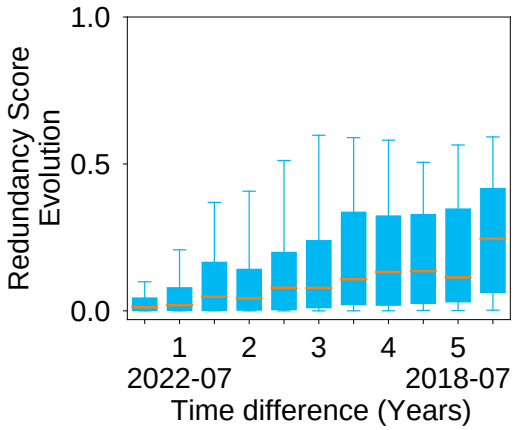


Figure 5.5: Evolution of the redundancy scores between two runs of MVP. After one year, the evolution of the pairwise redundancy scores starts to significantly increase.

Unsurprisingly, the higher the value of m, the greater the pairwise redundancy score evolution. As routing dynamics evolve over time, the partial view of each VP changes, capturing different portions of the Internet. For example, if an AS hosting a VP changes its economic relationship with one of its peers, this VP will see new routes which alters its partial view, due to the routing policies being updated subsequently. However, these changes are not significant when $m < 12$, as we observe that the pairwise redundancy score evolutions remain low, below 0.1. We find that the redundancy scores change by less than 5% between two runs of MVP spaced by 12 months or less. After one year, the evolutions start to significantly increase, going from 0.07 after one year, to 0.13 after two years, almost doubling. The differences can be as high as 0.32 in the median

case when $m = 66$. We calibrate GILL to recompute the set of anchor VPs every year, as it appears to be the best tradeoff between stability and computational expense.

Observe that GILL's operators can temporarily override these default parameters to accommodate bursts of newly deployed VPs, e.g., when the platform is bootstrapped with another BGP data collection platform like RIS LIVE or RV BMP Live. In this section, we explained how GILL works fundamentally. We now present the implementation of all GILL's components. A prototype of this platform is currently up and running at https://bgproutes.io.

5.3 System implementation

In this section, we describe the implementation of GILL. A prototype of GILL is currently running at https://bgproutes.io and collects real BGP data from several peers. To provide a head start in visibility, we bootstrapped GILL with RIS and RV peering sessions, allowing it to collect data from $\approx$ 2500 peers. The system comprises three main components: a custom BGP daemon optimized to collect BGP data, an orchestrator that manages all required tasks (e.g., setting up new peers or recomputing the filters), and a web interface that provides diverse tools such as looking glasses and a form that enables operators to set up a peering session with GILL.

We start in §5.3.1 by providing an overview of the system, its components, and how they interact with each other. Next, we present the implementation details of our custom BGP daemon used to collect data from GILL's peering sessions (§5.3.2). Then, we introduce in §5.3.3 the orchestrator and all its features. In §5.3.4, we describe the web interface, the available tools, as well as the peering session establishment process automation. We then explain in §5.3.5 how we used the data from RIS and RV to bootstrap GILL. Finally, in §5.3.6, we conduct a performance evaluation of the system, demonstrating GILL's ability to scale up to tens of thousands of VPs.

5.3.1 Overview of the system

Custom BGP daemon. GILL operates within a container-based system, where each BGP daemon runs in a separate Docker container [145]. The overall system architecture is depicted in Fig. 5.6. Each BGP daemon in the Docker cluster acts as a BGP route collector, maintaining a peering session with one VP. All the BGP daemons share the same Docker network, enabling an automatic attribution of the IP addresses when new peering sessions are configured. Since the BGP collectors use remote peering sessions with VPs, they need access to the public Internet. This is facilitated by a MASQUERADE forwarding rule configured on the host server, enabling any of the services running on the docker network to access the public Internet. Each BGP daemon collects BGP

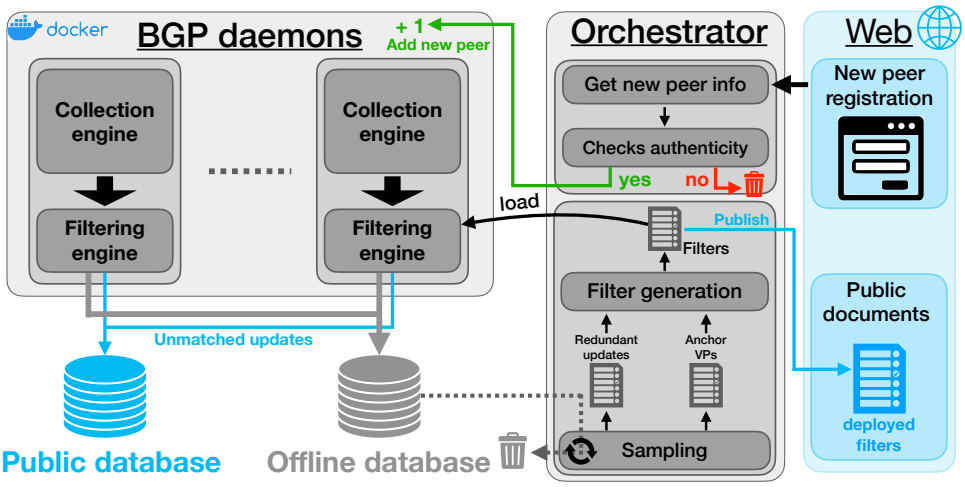


Figure 5.6: GILL's system workflow. Public components are depicted in blue, while internal components are depicted in gray.

data from one VP and writes the data in two separate databases, using MRT format [85]. The first database is the *public database* where the retained updates (i.e., those that have not been filtered) are dumped. The second is the *offline database*, where all updates are dumped, whether they match the filters or not. The offline database is not accessible to users and is used exclusively by the orchestrator to recompute the filters.

A task orchestrator. The orchestrator runs directly on the host server (i.e., outside of Docker containers), managing various critical tasks. It is responsible for running the filter recomputation process, pushing the new results in the filter engine of each corresponding BGP daemon, and publishing the new filters on the web interface. The orchestrator uses the offline database to recompute the filters, subsequently discarding the database once the process is completed. The data in the offline database is stored for 30 days when recomputing the anchor VPs, and for two days when recomputing the filters. Additionally, the orchestrator is in charge of setting up a new BGP collector (i.e., BGP daemon) when a new VP is added using the web interface.

A web interface. An operator willing to contribute to GILL by sharing its routes can fill out a form and after a brief security check detailed in §5.3.4, the orchestrator automatically initiates the creation of a new BGP collector. The web interface publishes various information, such as the list of peers contributing to GILL, their status, the volume of data collected, and the filters currently applied to each specific peer. Additionally, the web interface provides a *looking-glass* service that enables users to see the current state of the RIB for any active VP in real-time.

We now give the implementation details of all the three system components, starting with the custom BGP daemon.

5.3.2 GILL's custom BGP daemon

Current BGP data collection platforms use existing BGP daemon to collect data from their peers. For each route collector, they dedicate either one server, on which they run software routers (e.g., FRRouting for RV) or a hardware router. Platform operators configure multiple sessions on route collectors, one for each VP. The collected updates and the RIBs (i.e., a snapshot of the routes), are dumped at regular intervals, in MRT format. For instance, RV dumps BGP updates every 15 minutes and RIBs every two hours. However, these solutions are not optimized for data collection, particularly to implement an overshoot-and-discard collection strategy. As they build a human-readable RIB, routers (softawre or hardware) transform each collected update into an intermediate data structure. While this step helps operators debug received routes, this process is resource-consuming and unnecessary in the context of BGP data collection. Moreover, although these solutions often implement the filtering process with *route-maps*, they cannot scale to hundreds of thousands of different filters, even at a coarse granularity. We demonstrated this by trying to deploy n filters on a FRRouting router running on a 24 cores Intel(R) Xeon(R) W-2265 CPU 3.50GHz CPU for $n \in \{1, 10, 100, 1000, 10000, 100000\}$ and report the maximum number of filters that the router can handle. We configured filters that matched only the prefix contained in the BGP updates. The highest value for n where the router did not crash is 1000. However, because GILL builds filters that aim at discarding 95% of the data, GILL may push hundreds of thousands of filters for each VP. For all these reasons, we decided to build our own custom BGP daemon, optimized for data collection, removing all non-essential features.

The custom BGP daemon requires significant processing performances when parsing the received BGP updates, filtering them, and writing them on the disk. Consequently, we implemented it using C language, as compiled programs exhibit better processing performance. A comparison with an implementation made in Python showed that the Python implementation was $\approx$ three times slower, due to less optimized operations on byte arrays. Our BGP daemon comprises two main engines: one collection engine and one filtering engine. The workflow of the custom BGP daemon is illustrated in Fig. 5.7.

Data collection engine. The data collection engine is responsible for collecting the BGP updates from a VP, maintaining the BGP session, and opening new dump files. Upon launch, it starts by reading a configuration file containing information about GILL and the peer (i.e., VP): GILL's ASN, GILL's public IP, peer's ASN, and peer's public IP. Using this information, it creates a socket and initiates a TCP connection with the

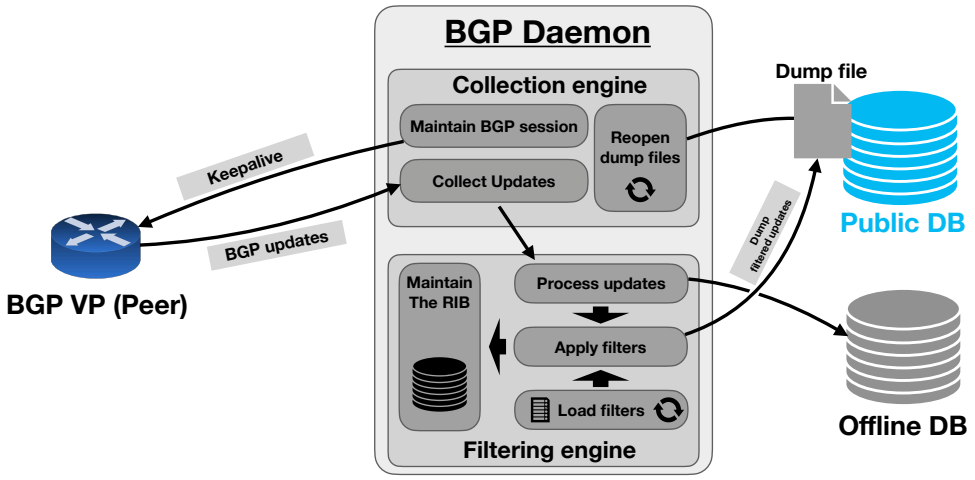


Figure 5.7: Workflow of the custom BGP daemon. The two main components are a collection engine and a filtering engine.

peer, on port 179 (the default BGP port). If the connection is not successful, the daemon starts a recurrent background task that attempts a TCP connection every 60 seconds until the TCP session is established.

Once the TCP connection is successful, the BGP daemon starts the BGP session by sending an OPEN message to the peer, containing information about GILL, such as the ASN (read from the configuration file). Upon receiving back an OPEN message from the peer, the daemon checks the consistency of the information sent with the information read from the configuration file (i.e., the IP address or the ASN). If all the checked information matches, the BGP session is established upon receiving the first KEEPALIVE message.

Once the BGP session is established, our BGP daemon runs two different tasks. The first task aims at maintaining the BGP session. An essential information sent in the OPEN message is the *hold timer*, which defines the time before a router disconnects the BGP session if no BGP messages are received. Since our BGP daemon does not send or forward any routes, the peer will not receive any messages. Therefore, the daemon runs a background task in charge of sending a KEEPALIVE message every "hold timer"/3 seconds, as suggested in the BGP RFC [1]. This process ensures the BGP session is maintained. Conversely, if no message is received from the BGP daemon peer before the hold timer expires, it disconnects the TCP connection.

The second task run by the collection engine upon establishment of the BGP session is the recurrent reopening of the dump file. We dump the updates received by a given BGP daemon in a folder whose name corresponds to the ASN and the public IP of the peer.

However, dumping all updates in the same file might cause sub-optimal processing issues. For instance, if a user wants to restrict their analysis to a specific time period, they would still need to process the entire file. Consequently, similar to what RIS and RV do, we separate the dumps into different files, grouping the updates that appear in the same time range. Our BGP daemon opens a new dump file every five minutes. Observer that this default parameter can be overridden.

Data filtering engine. The data filtering engine is responsible for loading the filters, filtering the updates, dumping them into the database, and maintaining the RIB. Every 24 hours, the BGP daemon scans for changes in the filters. The computed filters are sent to the Docker container by the orchestrator in the form of a file, where each line corresponds to a prefix that must be filtered. If any changes in the filters are detected, the BGP daemon withdraws the current filters and loads the new ones using a radix tree. This data structure significantly reduces the complexity of searching for an element, requiring at most n bit-to-bit comparisons, where n is the bit length of the longest element stored in the tree, which is 128 bits at maximum in our case. During the process of loading the filters, all received updates are temporarily queued and processed once the new filters are loaded. This ensures that no updates are lost during the filter update process.

When the BGP daemon receives updates, it discards the ones that match any of the filters, i.e., those for which the prefix is loaded in the radix tree. The retained updates are dumped into the current dump file in MRT format. It also updates the RIB to maintain the up-to-date state for each prefix. The RIB is a hash table where the key is a prefix, and the associated value is the binary version of the corresponding BGP attributes (i.e., the raw attributes carried in the update, in binary format).

The orchestrator requires all the BGP data to recompute the filters, including filtered updates. When the filters need to be updated, the orchestrator sends a specific signal to each BGP daemon. Upon reception of such a signal, the BGP daemon dumps all the received BGP updates (even if they match the filters) to temporary files in the offline database. Observe that the updates that do not match any of the filters continue to be dumped into the public database. For both the public and the offline databases, the data is dumped on a shared directory, implemented by Docker volumes. This setup facilitates file exchange between the docker container hosting the BGP daemon and the host server without additional processing from the daemon. The data from all the BGP daemons is gathered into a directory that represents either the public or the offline database.

To interact with the daemon, other components of the system use a custom remote shell that enables sending commands to the BGP daemon. The remote shell is also included in the Docker container of each BGP daemon.

Custom remote shell. The custom remote shell is responsible for sending commands to the BGP daemon for various purposes such as restarting the daemon, retrieving the RIB, or activating the dump in the offline database. It connects to the BGP daemon using a dedicated socket and sends the commands input by the user. An option also enables one to write the command to execute directly on the command line. The commands are processed by the BGP daemon and the responses are sent to the remote shell through the same socket. This feature enables the web interface to implement looking glasses by accessing the RIB in real-time.

5.3.3 GILL's orchestrator

GILL's orchestrator manages multiple tasks: setting up a new BGP collector (i.e., BGP daemon) upon receiving new peering requests from the website, recomputing correlations between updates, updating the anchor VPs, and generating the filters. To communicate with the BGP daemons, the orchestrator uses two different processes. It either sends a signal to the daemon by using the custom remote shell or transfers files using the docker volume which facilitates sharing files between the BGP collectors and the server hosting the docker environment. The orchestrator is implemented in Python, as it does not require high processing performances. All implemented algorithms effectively scale with their computation frequency.

Setting up new BGP collectors. Establishing a peering session between a VP and a BGP collection platform is often a very time-consuming process. Discussions with BGP collection platform operators confirmed that this process typically involves exchanges of information using mail and requires human processing. To overcome this limitation, GILL proposes an automated peering session establishment process. Using the web interface, the operators willing to contribute to GILL can register their ASN and public IP to automatically configure a BGP collector on GILL's side. The orchestrator then performs two actions: verifying the authenticity of the VP and setting up the BGP collector. Given that anyone, including potential malicious actors, can attempt to configure a BGP collector (e.g., to pollute the collected data), the orchestrator implements a security check. The operator registering the new VP has to send a mail (with empty content) to the address *peering@bgproutes.io*. In order to ensure the legitimacy of the email sender, the orchestrator retrieves registered AS's email address from PeeringDB [83] and only validates the VP's authenticity if the received email originates from this address. This process ensures that the operator configuring the BGP collector legitimately operates the registered AS. Once the security check is satisfied, the operator uses the information

gathered from the form to create a new BGP collector configuration file, activate a new Docker container, and start the BGP daemon using the appropriate configuration.

Redundant update computation. The filters deployed in each BGP collector depend on two pieces of information: the redundant updates and the anchor VPs. To recompute the redundant updates, the orchestrator uses the remote shell to send the following signal to each BGP collector: *"dump all received BGP updates in the offline database, even if they match any filter for the next 48 hours"*. The orchestrator requests for 48h of data as we demonstrated in §4.1.1 that capturing consistent correlations between updates requires two days of data. The data in the offline database is separated per prefix and per VP, i.e., every BGP collector dumps the data for each prefix in a different file, as the correlation groups are built per prefix. This per-prefix organization of the data enables a simpler parallelization of the process, saving a consequent amount of time. With our server equipped with 24 cores Intel(R) Xeon(R) W-2265 CPU 3.50GHz and 64GB of RAM, this process takes $\approx$ 16 hours when using data from all RIS and RV VPs. After collecting the data in the offline database, the orchestrator runs BUS to classify the redundant updates. It runs the correlation group construction (§4.1.1) and the reconstitution power algorithm (§4.1.2) in parallel for each prefix. Exploiting the prefix-wide redundancies (§4.1.3) is not feasible in parallel. However, as this process is not time-consuming, performing it sequentially does involve any performance issue. The orchestrator then updates the latest set of redundant updates with the new one.

Anchor VPs computation. The orchestrator initiates the anchor VPs computation process by sending the following signal to each BGP collector: *"Dump all received BGP updates to the offline database, whether they match a filter or not for the next 30 days"*. The orchestrator requests 30 days of data as this time frame is required by MVP to find events and assess the pairwise redundancy between VPs. Unlike the redundant update computation, the orchestrator does not separate the files in the database per prefix, as the events are not computed on a per-prefix basis. The orchestrator then runs MVP to *(i)* find a balanced set of BGP events, *(ii)* computes the topological features, *(iii)* computes the pairwise redundancy scores, and *(iv)* generates the set of anchor VPs. Using our server, this process takes $\approx$ 22 hours when using data from all RIS and RV VPs. This computation time scales with the yearly update rate of the anchor VPs. The orchestrator then updates the current set of anchor VPs with the new one.

Filters recomputation. Upon computing redundant updates or new anchor VPs, the orchestrator initiates GILL's process of filter computation, as described in §5.2. Once the filters are computed, the orchestrator writes the corresponding filters in the shared directory of each BGP collector. Since the BGP daemons scan for changes in the filters, after a maximum of 24 hours, the updated filters are pushed. Finally, the orchestrator publishes the updated filters on the web interface. This allows any operator wondering

why a given prefix does not appear in the collected updates or in the looking glass to determine whether the cause is the prefix being filtered or route propagation issues.

5.3.4 GILL's web interface

GILL's system includes a web interface that enables the user to interact directly with the platform. The website is accessible at https://bgproutes.io and provides different tools. First, it provides the list of GILL's peers (i.e., the VPs) with all related information. Second, it provides a looking-glass tool, available for every active VP. Third, it provides an HTTP database where the users can download any update files from any available VP. Fourth, it provides a form where the operators willing to contribute to GILL can register new VPs. Finally, it provides a tool that enables users to build their own set of anchor VPs on which they can force some characteristics (for instance forcing the presence of one or more VPs). The website is designed for responsiveness using HTML, CSS, JavScript, and Jinja2 to ensure a responsive website. The web server runs using the Flask python application, served with a Nginx web proxy.

VP list and information. The first component of our website displays the list of all VPs at the */peers* endpoint. The list of peers comprises five pieces of information: the AS name extracted from the as2org CAIDA dataset [146], the ASN, the peer IP, the date at which the VP was added, and the date at which it was removed. Each entry in the list is clickable, redirecting to another page where the user can get more specific information about the peer. This new page provides statistics about the peer, such as the time at which the BGP session has been established, the time at which the latest keepalive has been received, the number of received updates, the number of received prefixes, the number of filtered prefixes, the current size of the RIB, and the list of prefixes that are filtered on this VP. All this information is refreshed every 60 seconds to provide near-real-time statistics.

Looking glass. GILL's web interface provides a looking-glass tool for every active VP, accessible via the *looking glass* tab. This tool provides users a list of all peers with similar information as the list of peers available at the */peer* endpoint. However, when a user clicks on a given peer, it opens a new page displaying all the prefixes and their corresponding BGP attributes currently stored in that peer's RIB. As a web browser may not be capable of displaying the typical 1M entries stored in a RIB, by default a page only displays 100 entries and the user can navigate through additional pages to see other entries. When a user queries the looking glass for a given VP, the web server uses the custom remote shell of the corresponding BGP daemon to get the RIB, in JSON

format. If a peer becomes unavailable e.g., if the VP is rebooting, the looking glass for this peer is temporarily disabled.

Public database. GILL's web interface provides an HTTP server that publishes the directory containing all the BGP update dump files This enables the users to download any file, from any available VP. The database is available in the *database* tab and is served using a ruby HTTP server. The path to a given file is of the following form: http://bgproutes.io:5050/peerASN_peerIP/year/month/day/timestamp.mrt.bz2 This setup provides an easy and efficient way for users to access and retrieve BGP update files as needed.

New peer registration form. As mentioned in §5.3.3, GILL implements an automated process for configuring new BGP collectors, in order to limit human processing. On the website, an operator can access a form (on the *new_peer* tab), where he can register the ASN and the public IP of the router that will share routes with GILL. After filling out the form, the public peering information of GILL is presented to the operator, such as GILL's ASN and public IP. After the security check described in §5.3.3, the operator can configure the BGP session on the register router (which then becomes a VP) and after a maximum of 60 seconds, the new VP appears on the list of GILL's peers.

Anchor VPs customization. As presented in §2.3.2, users often resort to sample BGP data in an arbitrary fashion, resulting in suboptimal performances for many objectives. GILL addresses this issue by providing the list of anchor VPs through the web interface. This list enables any user who wants to sample the BGP data by selecting a subset of the VPs to use a set of least redundant VPs. However, some users might have some specific needs that require customizing the set of VPs. A user with higher processing power may also want to process data from more VPs than the ones used as anchor VPs. Therefore GILL's web interface proposes a tool to build a custom set of VPs, where any user can enter the number of required VPs, the maximum data processing budget he has (in terms of data volume), or include some VPs. This tool is available on the *custom_anchor* tab.

5.3.5 GILL's bootstrapping with RIS and RV data

GILL has the potential to significantly improve the state-of-the-art in BGP data collection platforms. However, this can only be achieved if it attracts a substantially larger number of VPs—an order of magnitude more than current platforms like RIS and RV. To enhance the attractiveness of peering with GILL, we bootstrap it using data from RIS and RV peering sessions. Specifically, we mirror all the peering sessions from RIS and RV ($\approx$ 2500 combined) and integrate them into GILL. This bootstrap enables a head start of visibility for GILL, making it already the public collection platform with

the highest number of (mirrored) peers! Since RIS and RV do not provide the same live-stream data tools, we developed two different proxies to leverage their respective peering sessions.

Leveraging RIS peering sessions. RIS provides data in real-time through its live-stream data tool [147], which operates using a websocket. Users can connect to the websocket and subscribe to one or more feeds. RIS live provides a separate feed for each of their 23 active route collectors. Upon subscribing, users receive each single collected BGP message (including the keepalive messages) in a JSON format. This format includes multiple information, such as the ASN and the public IP of the sending VP, the timestamp, and all the main BGP attributes. It also informs users of the status of any VP, by sending a *peer-disconnected* message every minute for each disconnected VP. This live-stream data tool is extensively used for real-time BGP anomaly detection. For instance, it is used in ARTEMIS [69], a tool that enables detecting BGP hijacks in real time.

This live-stream data tool provides feeds only at a per-collector granularity. However, GILL aims at filtering and providing data at a per-VP granularity. Additionally, we want the different types of peers (original or mirrored) to be transparent to the orchestrator and the filter recomputation process. Therefore, we built a proxy that takes the live data as input and feeds BGP daemons (as described in §5.3.2). The mirroring system for RIS peers uses three main components: a BGP daemon orchestrator, a proxy, and a cluster of BGP daemons. The architecture of the RIS peering session mirroring system is described in Fig. 5.8.

Live-stream data proxy: The proxy is responsible for establishing the connection with the websocket and retrieving the data stream. Once the stream starts, the proxy forwards the received messages to the corresponding BGP daemon. Over time, some new VPs may be added. Therefore, when the proxy receives a mirrored update for a VP that has no corresponding BGP daemon, it requests the BGP daemon orchestrator to set up a new BGP daemon. The messages for this new peer are temporarily queued until the proxy can connect to the new BGP daemon. While it has at least one message queued, the proxy attempts to connect to all the BGP daemons with at least one queued message every 10 seconds. Note that this parameter can be overridden. There are no synchronization issues since the timestamp at which the update was received is included in the message sent by the websocket. The BGP daemon thus uses this timestamp instead of the actual timestamp at which it receives the update. To parallelize the process of gathering live-stream data, we run one proxy for each of the 23 active route collectors deployed by RIS, each running on a different process and CPU.

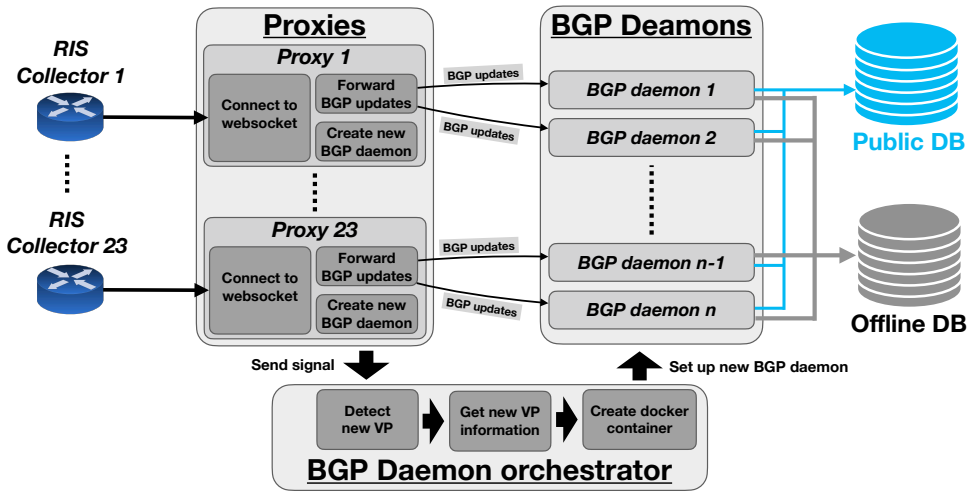


Figure 5.8: Workflow of the RIS peering sessions mirroring. The system comprises three components: a live-stream data proxy, a BGP daemon orchestrator, and a cluster of BGP daemons.

BGP daemon orchestrator: The BGP daemon orchestrator is responsible for setting up BGP daemons upon detecting a new VP by the proxy. This component runs as a separate process. When it receives a signal from any of the proxies, it retrieves the information relative to the new VP (ASN and public IP) and sets up a corresponding BGP daemon. The signal received from the proxy consists of the creation of a file in a shared directory. The file is created by the proxy and its name corresponds to the ASN and the public IP of the new VP. The BGP daemon orchestrator regularly scans the shared directory and creates a new BGP daemon when it detects the creation of a new file. Only one BGP daemon orchestrator is needed for all proxies, as the addition of new VPs does not occur frequently, thus not requiring specific performances.

BGP daemon cluster: The BGP daemons deployed by the BGP daemon orchestrator need to provide the same output as those deployed by GILL's system orchestrator. However, using a proxy to mirror data from RIS peers can lead to desynchronization of the timestamps due to forwarding delays. Additionally, queuing data received from a new peer can also cause synchronization issues. To address this issue, we developed a modified version of the former BGP daemon. Instead of receiving BGP messages directly, this daemon processes altered BGP messages, where the four first bytes correspond to the original timestamp, and the remaining bytes are used to store the original BGP message. This process allows us to embed the original timestamp of the BGP update into the message, removing the risk of timestamp desynchronization. This is feasible

since the original timestamp at which the BGP message is received by the RIS collector is contained in the JSON messages sent by the websocket.

Leveraging RV peering sessions. Unlike RIS, RV does not provide a websocket for a live stream of BGP updates. For each of their 43 collectors, they provide a live BMP [105] feed. However, at the time this thesis is written, no tool can effectively parse and collect this data. For instance, while BGPStream provides support for RV BMP live data, it quickly becomes desynchronized, with one second of data received roughly every three seconds. Therefore, we cannot rely directly on the BMP feeds. To leverage RV peering sessions, we build a pseudo-real-time proxy to feed the BGP daemons.

Instead of relying directly on a live stream data feed, our RV proxy implements a scheduler that collects the latest update dump file (in MRT format) published by RV in their database every 15 minutes. The proxy starts by downloading the update file and extracting the timestamp, the VP's ASN and public IP, and the raw update from each MRT entry. It then builds a custom BGP update with the four first bytes being the timestamp (as explained for RIS peering sessions above) and sends it to the corresponding BGP daemon. Similarly to the RIS proxy, if the BGP daemon is not up, the updates are queued until the BGP daemon orchestrator sets up the required container. Although we need to wait 15 minutes to get the data, this is the best we can do for now, despite this process not being real-time. Note that there is one such proxy for each of the 43 collectors maintained by RV.

5.3.6 GILL system's performances

We now evaluate GILL's system implementation to ensure that it can scale to tens of thousands of VPs without requiring significant investments in hardware infrastructure. The critical point of GILL's system is the custom BGP daemon. We evaluate the capability of the collection and filtering engines to cope with the volume of data induced by a drastic increase in the number of VPs.

Experiment setup. We evaluate the maximum data rate that our BGP daemon can handle by configuring a set of X BGP daemons (with $X \in \{100, 1000, 10000\}$). For each running BGP daemon, we configured a fake peer that establishes a BGP session with the BGP daemon and sends a stream of BGP updates. The fake peers send bursts of BGP updates that match the data rate observed from RIS and RV dumps: it either sends updates at a frequency of 28K updates per hour (the average observed from RIS and RV sessions) or 241K updates per hour (the 99th observed percentile). We present the results of our BGP daemon's performances without filters and with filters that discard 94% of the updates, to align with the data reduction proposed by GILL's filters. The

Number of peers		100	1000	10000
With filters (i.e., GILL)				
Update load (per hour)	Average (28K upd/h)	0%	0%	0%
	99th percentile (241K upd/h)	0%	0%	high
Without filters				
Update load (per hour)	Average (28K upd/h)	0%	0%	39%
	99th percentile (241K upd/h)	0%	32%	high

Table 6: Proportion of updates lost by our BGP daemons as a function of the update frequency when using only one CPU. A green cell means daemons cope with the update frequency (no updates are lost) whereas a red cell means daemons drop at least one update. When the number of lost updates cannot be precisely computed because the load is too high, we just label it as high and color it in red.

benchmark is run on a single 3.20 GHz Apple M1 Pro CPU with 16GB of RAM. Table 6 presents the percentage of updates lost across the different considered scenarios.

Benchmark results. We find that a single CPU successfully handles (i.e., losing no updates) up to 10k BGP daemons with the average update frequency and up to 1k daemons with a high update frequency (99th percentile) when using filters. Thus, we expect GILL to support tens of thousands of BGP sessions (even during peak times) on a server with many CPUs. While one might assume that applying filters would be a time-consuming process, potentially lowering the performance of the BGP daemon, our observations reveal the opposite. We observe that our BGP daemons can process more updates when using filters, as the most time-consuming task of our daemon is writing on the disk, which is significantly reduced when applying the filters.

We evaluated the software performances of GILL and show its ability to cope with tens of thousands VPs. We now evaluate the performances of GILL's sampling, demonstrating its ability to perform effectively for a wide range of objectives.

6

Performance evaluation and impact

In the previous chapter, we analyzed the performance of GILL's system implementation. In this chapter, we evaluate the performances of GILL's sampling on various objectives. Specifically, we analyze how effectively the updates retained by GILL's filters enable conducting some popular measurement analysis. This includes evaluating both the short-term and the long-term impact of GILL. Given that GILL has not yet gained traction, evaluating the long-term impact of GILL with real data is impossible, as we cannot precisely determine what we are missing from non-deployed VPs. Therefore, we rely on simulations to compare GILL's implementation of the overshoot-and-discard strategy against the current BGP data collection strategy on scenarios with different VP coverages.

We start this chapter by benchmarking GILL's sampling in §6.1, comparing it against the state-of-the-art for BGP data sampling strategies across five common objectives. We then demonstrate in §6.2 the long-term impact of GILL's sampling on three objectives, illustrating the benefits of using GILL compared to the current BGP data collection strategy once GILL gains traction. Next, in §6.3, we highlight the benefits of using GILL for sampling BGP data through two measurement studies. More precisely, we reproduce the experiments conducted in two papers extracted from the literature where authors sample BGP data and show that data sampled by GILL yields better results,

induced by an improved visibility over the Internet routes. Finally, we present the benefits of GILL on DFOH [82], a forged-origin hijack detection system that relies on GILL's anchor VPs to sample BGP data.

6.1 Benchmarking GILL's sampling

In this section, we evaluate the two main key requirements of GILL's sampling: providing an effective sampling and avoiding overfitting toward any definition of redundancy. We demonstrate that GILL's sampling improves the trade-off between data volume and inferred information compared to both current BGP data sampling schemes and sampling strategies optimized for specific objectives. We use five different use cases to evaluate the inferred information.

We start by presenting in §6.1.1 the experiment setup used to benchmark GILL's sampling. Next, we compare GILL's sampling against the state-of-the-art strategy for BGP data sampling in §6.1.2. This experiment also allows us to evaluate the soundness of the two main key ingredients of GILL's sampling: the selection of redundant updates and the use of anchor VPs. Finally, we demonstrate in §6.1.3 that GILL does not overfit toward any definition of redundancy.

6.1.1 Experiment setup

We start by presenting the use cases on which we evaluate GILL's sampling. We then present how we compare GILL against other baselines.

Describing the five use cases. We selected five use cases to evaluate GILL's sampling, extracted from the literature. We carefully picked these use cases such that each BGP attribute is required for at least one of them, ensuring that GILL does not overfit toward a given attribute. For instance, the *time* is required to detect transient events (use case *I*); the *prefix* is required to detect Multiple Origin ASes (MOAS) hijacks (use case *II*); the *AS path* is required to map the Internet topology (use case *III*); and the *community values* are required to detect action communities (use case *IV*) and unchanged-path updates (use case *V*). For each of these use cases, we process updates and RIBs collected by all RIS and RV VPs during 30 one-hour periods selected in Nov. 2023. We select one-hour periods to reduce the computational expense of these analysis, but we mitigate the bias induced by time sampling by selecting 30 non-overlapping periods. We used these 30 one-hour periods to extract all observed events corresponding to each five use cases. We now detail, for each use case, how we select the events on which we evaluate GILL's sampling:

I Transient paths detection: We define a Transient path as a BGP route that appears in the RIB of a VP for less than 100 seconds. These routes are often attributed to path

exploration [148], which typically occurs after a failure. When the primary path is unavailable, an AS experiencing an inter-domain outage is likely to receive new routes toward the affected prefixes from its other neighbors. Because a significant amount of routes are potentially received, the router hosting the VP may not receive the new best route immediately after the failure. Consequently, routes installed in the RIB by the router are likely to be replaced quickly after, resulting in routes being installed in the RIB for a short timeframe. Transient paths can also occur when a router hosting a VP reboots, generating a similar phenomenon as for an inter-domain failure. Detecting transient paths can be useful for mapping hidden portions of the AS-level topology since transient paths often consist of backup paths that are typically not exported to route collectors. Additionally, transient paths can help researchers to better understand the dynamic inter-domain traffic engineering implemented by large content providers to balance the load received by their infrastructure [84, 149, 150].

To extract all the transient paths from the 30 one-hour periods, we start by building for each VP the RIB state at the beginning of each selected one-hour period. We build the RIB state for a given period by collecting the last RIB dump before the beginning of the period and completing it with the subsequent updates. We then process each one-hour period by updating the RIB with the corresponding updates. Each time a RIB entry is withdrawn by a new update (implicitly or not), we add the previous route in the set of transient paths if it has been present in the RIB for less than 100 seconds (We use 100 seconds to accommodate typical BGP convergence delays [42]). A transient path is identified by the timestamp at which the route was collected, the prefix of the route, and its origin AS. We do not use the entire AS path since redundant VPs might observe the same transient path (i.e., induced by the same BGP dynamic), but with different AS paths. We end up selecting 859K events over the 30 one-hour periods.

II MOAS hijack detection: Multiple Origin ASes (MOAS) hijack is a BGP event (legitimate or not) where two or more distinct ASes announce the exact same prefix. As a result, other ASes in the routing system will choose between one of the different available routes and some of them will likely select the illegitimate one (when the MOAS is malicious). When legitimate, MOAS can be due to prefixes associated to IXPs, multihomed ASes [151], or multinational companies owning multiple ASes and changing their prefix affectations [152]. When illegitimate, MOAS hijacks are used to divert the traffic from its legitimate destination [62, 153, 154]. This can be used to either analyze the hijacked traffic or blackhole it, disrupting the connectivity toward the hijacked prefix. For this use case, we only focus on illegitimate hijacks and use the methodology proposed by [68] to eliminate the large portion of false positives raised by the public hijack detection platforms.

To extract all the MOAS hijacks that occurred within the 30 one-hour periods, we start by building a prefix to AS mapping using the CAIDA pfx2as dataset [155] from Nov. 2023. Then, for each one-hour period, we process every single update by checking the consistency between the *prefix/origin AS* pair announced in the update and the one present in the CAIDA dataset. If the announced prefix is not in the CAIDA dataset, we do consider it as legitimate. If there is an inconsistency, we add the *prefix/origin AS* pair extracted from the update to the set of potential MOAS hijacks. An event is identified by the timestamp of the underlying update, its prefix, and the presumed illegitimate origin AS. To remove false positives from the set of potential MOAS hijacks, we apply the four filters described in [68] on each event. First, we eliminate all the events that match a valid RPKI record [156], using the RPKI database from NTT [157]. Second, we eliminate all events that match at least one entry from the IRR database [158]. Third, we apply the topology filter proposed by [65], which aims at removing events where the victim AS of the MOAS hijack is the provider of the attacker, as it is likely to be a legitimate behavior. Finally, we applied time filtering, as most of the illegitimate MOAS events are short-lived. We use the same settings as [68]. After applying all these filters, we end up with 1587 events when combining the 30 one-hour periods.

III AS topology mapping: Mapping the AS topology has been an extensively studied topic over the last two decades. This use case is often achieved by extracting AS links from collected AS paths. Additionally, some techniques leverage active measurement coupled with an IP-to-ASN mapping to provide additional AS links that cannot be observed from the BGP VPs [27, 28, 29, 30]. Effectively mapping the topology requires collecting as many diverse AS paths as possible, as it increases the likelihood of observing new links. Path diversity is also crucial for other AS topology mapping-related measurement studies, such as inferring AS relationships [35] or predicting AS paths [159, 160]. Some studies about mapping topology or inferring AS relationships focus only on stable paths [35], i.e., paths that appear regularly on large timeframes in the RIBs. However, for this use case, we choose to focus on both stable and transient paths, as transient paths can offer valuable information about the backup links in the Internet topology.

To extract both stable and transient links, we start by building the RIB states at the beginning of each one-hour period used for the benchmark, using the same methodology as for use case *I*. We then extract all the AS links from the collected AS paths, which we add to the set of observed AS links. To process transient paths, we use the updates collected during the one-hour period and add all the observed AS links to the set of AS links. We repeat this process for each of the 30 one-hour periods. We also handle AS paths with AS aggregation by de-aggregating the AS paths into all the possible combinations of AS paths. After processing the 30 one-hour periods, we focus on the 687K observed AS links.

IV Action communities detection: BGP communities have been extensively studied over the years, with many efforts aiming at better understanding their purpose and, more generally, providing a better understanding of the Internet routing ecosystem. Krenc et al. [20] provide a methodology to classify communities into two categories: *information* communities and *action* communities. They rely on a clustering algorithm, as well as a new metric, the *on-path:off-path ratio*, to classify the communities. This metric measures the ratio between the number of updates where a given community *asn:comm* appears and *asn* appears in the AS path of this update, versus the number of updates where *asn:comm* appears but *asn* does not appear in the AS path. Since action communities aim at influencing path selection, they are more likely to appear off-path, unlike information communities that are used to tag a route. Therefore, we focus on action communities, as they are more challenging to observe.

We extract the action communities from the updates collected during the 30 one-hour periods. Krenc et al. [20] provide a dictionary of inferred BGP communities with more than 60k entries, where each community value is associated with its classification. For each collected update, we parse the set of associated community values and add a community only if it is present in the dictionary and classified as an action community. Consequently, we focus on the 8683 action communities observed during the 30 one-hour periods.

V Unchanged-path updates detection: A lot of noise often occurs on the Internet [161] and pollutes BGP data. This noise can be attributed to many events, such as persistent session restarts either from the BGP collector [162] or from other ASes in the topology [163]. Another definition of noise can be the proportion of duplicated announcements [164], i.e., receiving a BGP update for a prefix where all BGP attributes exactly match the one stored in the RIB for the same prefix. For this use case, we focus on another definition of noise, the *unchanged-path updates* [22], where a router receives an update for which only the set of associated BGP communities change. This new update, while different from the one stored in the RIB, does not signal any forwarding change. Recall that, if we choose to focus on this definition of noise, there exist many more.

To extract all unchanged-path updates, we start by building the RIBS for all 30 one-hour periods and for all VPs, using the same methodology as described for use case *I*. We then process every single update received during the 30 one-hour periods and compare them to the corresponding entry in the RIB. If the AS path is the same but the set of BGP communities has changed, we add this update to the set of considered unchanged-path updates. An unchanged-path update in this set is identified by its timestamp, prefix,

and origin AS. Consequently, we focus on the 263k unchanged-path updates collected from the 30 one-hour periods.

Comparing the different baselines. We benchmark GILL's sampling to demonstrate its effectiveness in improving the tradeoff between data volume and the information inferred across the five use cases described above. Specifically, we compare GILL's sampling against four naive baselines and eight *specific* baselines optimized for a given definition of redundancy. We start the comparison by computing GILL's filters in Oct. 2023. For each 30 one-hour period, we filter the collected updates and consider only the ones that are retained to infer the collected information. We then compute the volume of retained data, i.e., the number of retained updates. Then, we parameter every other baseline to collect the exact same volume of data as GILL, and use the retained data to infer the collected information. Next, we present, for each use case, the proportion of events or AS links that we can detect using the data retained by each baseline. If a baseline b has a score of 90% for use case *III*, it means that using the data sampled by b, we are able to map 90% of the AS-level topology (i.e., we observe 90% of the AS links). With the default parameters, GILL's sampling retains 6.7% of the updates collected during the 30 one-hour periods. Therefore, we configure the other baselines to retain the same proportion of updates, enabling a fair comparison between all the baselines, such that any performance gap can be confidently attributed to GILL.

6.1.2 GILL performs better than current techniques

In this section, we compare GILL's sampling against four naive baselines, which represent the state-of-the-art for BGP data sampling strategies. Additionally, we compare GILL against two modified versions of itself, in order to validate the two key design choices of GILL's sampling strategy. We start this section by describing the naive baselines.

Naive baseline description. We compare GILL's sampling against four naive baselines that sample BGP data either at the update or the VP granularity: *Rnd.-Upd.*, *Rnd.-VP.*, *AS-Dist.*, and *Unbiased*.

Rnd.-Upd.: This baseline relies on an overshoot-and-discard strategy by generating filters that are applied to VPs. Instead of generating filters based on the reconstitution power or by selecting anchor VPs akin to GILL, using reconstitution power, this baseline generates filters randomly. Filters are iteratively built by randomly selecting a VP and a prefix (using a uniform distribution) and adding it to the set of randomly generated filters. We stop adding new filters in the set when the selected filters enable collecting the same number of updates as GILL. This baseline is used to highlight the effectiveness of GILL's filter generation pipeline.

Rnd.-VP: This baseline samples BGP data by selecting a subset of the VPs. It generates a set of VP by iteratively adding a random VP until the set of selected VP collects the same number of BGP updates as GILL. According to our survey, this is the most widely used strategy to sample BGP data. This baseline is designed to compare the relevance of sampling BGP data at the update granularity against sampling at the VP granularity.

AS-Dist.: This baseline samples BGP data by selecting a subset of the VPs, but based on their AS distance, unlike *Rnd.-VP* where the VPs are selected randomly. In §3, we demonstrated that, despite being intuitive, the AS distance is not an effective metric for assessing redundancies between VPs. We use the *AS-Dist.* baseline to experimentally illustrate that this strategy may fail for multiple objectives. To implement this baseline, we start by computing the AS distance between all pairs of existing VPs. We initialize the set of VP with a random VP and then iteratively add the VP that maximizes the average AS distance from the already selected VPs. We stop adding new VPs when the number of updates collected by the selected VPs matches the number of updates retained by GILL. This baseline aims at illustrating the limitations of using naive metrics to assess the redundancy between VPs.

Unbiased: This baseline samples BGP data by selecting a subset of the VPs that minimizes the bias. In §3, we argue that selecting VPs based on their geographical positions or topological properties (i.e., position in the AS-level topology) may fail for a wide range of objectives. In [118], Sermpezis et al. defined the bias in a set of BGP VPs using various metrics such as the customer cone size of the AS hosting the VP or the country in which the AS is registered. We implement this baseline by using this definition of bias and build a set of VPs that minimizes the bias, thereby maximizing the diversity of the route collectors. We initialize a set of VPs containing all available VPs and iteratively remove the VP that most increases the *overall bias* (as defined in [118]) of the set. We stop removing VPs when the set of VPs collects the same number of updates as GILL. Similarly to *AS-Dist.*, this baseline aims at illustrating the limitations of using naive metrics to assess the redundancy between VPs.

GILL modified baseline description. Additionally, we compare GILL's sampling against two modified versions of GILL, *GILL-upd.* and *GILL-vp.* This comparison enables us to evaluate the soundness of the two key design choices of GILL, namely " Support for a flexible definition of redundancy" and "Retain all updates from a few valuable VPs".

GILL-upd.: This baseline consists of GILL's generated filters, but incorporates only the first key ingredient. Specifically, it computes the set of redundant updates, but without selecting any anchor VPs. Consequently, this baseline is likely to retain all the data from none of the available VPs. We adjusted GILL's default parameters to ensure that

the number of updates retained by *GILL-upd.* matches the number of updates retained by GILL. This adjustment ensures a fair comparison between the baselines.

GILL-vp: This baseline consists of GILL's generated filters, but incorporates only the second key ingredient. Specifically, it computes a set of anchor VP using the computed redundancy scores but does not apply any filters to any VP. We adjust GILL's default parameter, ensuring that the number of updates collected by the *extended* set of selected anchor VPs matches the number of updates retained by GILL. This adjustment ensures a fair comparison between the baselines.

	Sampling Scheme	GILL	GILL-simplified		Naive			
			GILL-upd	GILL-vp	Rnd. Upd.	Curr.	AS-Dist.	Unbiased
Use cases	**Trans. path detection (*I*)**	96%	65%	75%	83%	75%	95%	89%
	MOAS detection (*II*)	95%	95%	45%	33%	35%	59%	46%
	Topo mapping (*III*)	90%	93%	61%	72%	41%	67%	38%
	Action Coms. detection (*IV*)	91%	95%	49%	79%	48%	41%	42%
	Unchanged-path Upd. detection (*V*)	87%	60%	80%	76%	74%	43%	61%

Table 7: GILL's sampling outperforms all naive baselines, for all use cases. The color means that GILL performs better (), worse (), or similarly () than the baseline.

Table 7 presents the results of this benchmark, detailing the performances for all baselines across all five objectives. Each cell contains a number representing the proportion of events (or links) related to the use case on the x-axis that can be observed using the sampling strategy implemented by the baseline on the y-axis. The color of a cell is green if GILL outperforms the baseline in the y-axis for the use case in the x-axis, red if the baseline performs better than GILL, and yellow if the baseline and GILL perform similarly ($\pm$5%). The cells in GILL's column are colored in gray, as it serves as a reference to compare against other baselines. From these results, we can infer the two following takeaways:

Takeaway #1: GILL outperforms all naive baselines across all possible objectives and sometimes significantly. GILL detects +62%, +60%, +36%, and +49% MOAS hijacks (use case *II*) compared to *Rnd.-Upd.*, *Rnd.-VPs*, *AS-Dist.* and *Unbiased*, respectively. The only scenario where a naive baseline performs similarly to GILL is when we use the *AS-Dist.* baseline to detect Transient paths. These results highlight the suboptimality of the current BGP data sampling strategies and demonstrate the ability of GILL's sampling to focus only on the most valuable portion of the data.

Takeaway #2: The two key ingredients of GILL are relevant. *GILL-upd.*, which implements the first key ingredient, outperforms *GILL-vp* for use cases *II*, *III*, and *IV*, as it collects more diverse BGP attributes. Conversely, *GILL-vp* outperforms *GILL-upd.* for use cases that require all prefixes but not a high link visibility, namely *I* and *V*. *GILL-vp* performs better in this case because it retains all prefixes, even if redundant. Unlike its modified versions, GILL performs effectively for every use case, highlighting the relevance of combining the two key ingredients.

6.1.3 GILL does not overfit

In this section, we compare GILL's sampling against *specific* baselines. We show that, unlike these baselines, GILL does not overfit toward any use case nor any definition of redundancy. We start by describing the different baselines used to evaluate GILL's sampling.

Use-case-based specifics description. These baselines sample BGP data at the update granularity. For each of the five use cases described in §6.1.1, we build a *use-case-based specific* baseline that collects only the updates that enable effectively conducting the corresponding use case. We greedily add to the set of processed updates the one that enables the more information inference. For instance, if we consider use case *III*, we iteratively add the update with the AS path that enables discovering the most AS links not observed in previously processed AS paths. If we obtain a set of updates that enables detecting 100% of the events (or links) but contains fewer updates than those retained by GILL, we randomly add updates until the volumes match. These five baselines aim at striking the best tradeoff between the volume of collected data and the inferred information according to a given objective.

Definition-based specifics. These baselines sample BGP data by selecting a subset of the VPs. We build three *definition-based specific* baselines, each optimized for one of the three definitions presented in §3.2. We greedily build a set of VPs whose collective set of collected updates minimizes the redundancy according to one of the three per-update redundancy definitions. We iteratively add the VP that least increases the overall redundancy in the set of collected updates. We stop adding new VPs when the number of collected updates matches the number of updates retained by GILL, ensuring a fair comparison between the baselines.

Table 8 presents the results of the second part of the benchmark, formatted similarly to Table 7. The number in a cell represents how effectively a baseline performs for a given objective and the colors follow the same conventions as for Table 7. For ease of understanding, we also include GILL's sampling results, even though they are presented in Table 7. From these results, we can infer the two following takeaways:

	Sampling Scheme	GILL	Definition-based specifics			Use-case-based specifics				
			Def. 1	Def. 2	Def. 3	*I*	*II*	*III*	*IV*	*V*
Use cases	**Trans. path detection (*I*)**	96%	76%	98%	82%	100%	83%	79%	82%	83%
	MOAS detection (*II*)	95%	47%	48%	40%	36%	100%	32%	33%	32%
	Topo mapping (*III*)	90%	43%	49%	33%	72%	72%	100%	72%	64%
	Action Coms. detection (*IV*)	91%	46%	45%	44%	78%	77%	85%	100%	73%
	Unchanged-path Upd. detection (*V*)	87%	63%	63%	90%	76%	76%	71%	76%	100%

Table 8: Unlike the *Use-case-based specifics* baselines, GILL's sampling avoids overfitting. The color means that GILL performs better (), worse (), or similarly () than the baseline.

Takeaway #1: The definition-based specific baselines perform poorly in almost all possible scenarios. There are only two scenarios where a definition-based specific performs similarly to GILL. The baseline optimized for Def. 2 performs well in detecting transient paths, as it maximizes path diversity. The baseline optimized for Def. 3 performs well in detecting unchanged-path updates, as it includes the community values in its definition. In all other scenarios, GILL performs significantly better. For instance, GILL detects +45%, +46%, and +47% action communities (use case *V*) compared to the specific optimized for Def. 1, 2, and 3, respectively. These results demonstrate the soundness of GILL's filter generation and highlight that specific definitions of redundancy often fail when applied to practical use cases.

Takeaway #2: GILL's sampling generalizes effectively, whereas *use-case-based specific* sampling strategies tend to overfit. A *use-case-based specific* baseline performs effectively for the use case it is optimized for (100% of the objective is achievable for every use case), but performs poorly for other objectives. Consequently, GILL's sampling strategy outperforms every *use-case-based specific* baselines for the use cases they do not optimize. For instance, the *use-case-based specific* optimized for use case *III* enables observing +10% AS links compared to GILL, but GILL enables observing +18%, +18%, +18%, and +26% AS links compared to *use-case-based specific* optimized for use case *I*, *II*, *IV*, and *V* respectively. These results highlight the robustness of GILL's sampling algorithm, as it does not overfit toward any use case. They also demonstrate that overfitting toward a given use case results in poor performances for other use cases.

6.2 Long-term impact of GILL's sampling

The long-term impact of GILL will only become visible when it has thousands of sessions with VPs. Since it is challenging to know what is actually missed from VPs that are not deployed, we instead rely on controlled simulations to evaluate GILL in scenarios with a larger VP coverage. We generate a *pruned known AS topology*

and ten *artificial topologies* using the methodology described in §2.1.3, based on the C-BGP simulator [107]. We restrict our simulations to 1k-AS topologies, to reduce the computational resource requirements.

Use cases. We evaluate the long-term impact of GILL using the three use cases presented in §2.1.3, namely Topology mapping, Failure localization, and Forged-Origin Hijack detection. For both topology mapping and failure localization, we focus on the p2p links, as they are the most challenging to observe. For the forged-origin hijack detection use case, we focus on type-1 hijacks, as they are the most commonly used by attackers [82]. For each of these use cases, we generated events using the same methodology as described in §2.1.3.

Baselines. We compare GILL's sampling against two baselines that both represent possible BGP data collection strategies, the *current* approach, and the *best* approach.

Current (Curr.): This baseline consists of building a subset of the deployed VPs whose number of collected updates matches the one collected by GILL. This ensures a fair comparison with GILL, as both baselines collect the same volume of data. We select the VPs randomly, as it appears to be the common behavior when selecting VPs, according to our survey (§2.3.2).

Best: This baseline represents the ideal scenario, where we collect **all** the data from **all** available VPs, even with a high VP coverage. We then use all the available data to infer information for the three use cases. Consequently, this baseline does not enable a fair comparison with GILL, as it collects significantly more data. However, it provides an upper bound of the possible performances that a system like GILL can achieve.

Simulation settings. We tested different VP coverages (i.e., the proportion of ASes in the routing system hosting at least one VP) ranging from 2% to 50%. We choose 2% as the lower bound as it corresponds to the rounded-up VP coverage observed from RIS and RV infrastructures. Since GILL is data-driven, it requires past BGP updates to evaluate the correlation between updates triggered by BGP events. Therefore, we simulate 500 link failures (distinct from those used for the *failure localization* use case) and feed GILL with the induced updates, collected by every deployed VP (depending on the VP coverage of the corresponding scenario). The failures are generated by randomly selecting a link in the topology (with a uniform distribution). Each generated failure is separated by more than 100 seconds in time, as it is the time threshold implemented by GILL to build correlation groups, ensuring that each failure is a distinct event.

Simulation results Unlike §6.1 where we focused on a set of events observed in the RIS and RV data, we now have the ground truth. For each sampling scheme, we compute the proportion of events (or AS links) that each baseline detects among all

Coverage		2%			10%			25%			50%		
Data Collection Scheme		GILL	Curr.	Best	GILL	Curr.	Best	GILL	Curr.	Best	GILL	Curr.	Best
Upd. retained / Anchor VPs		*18.0% / 17.0%*		100%	*7.9% / 3.3%*		100%	*5.4% / 1.3%*		100%	*4.7% / 0.9%*		100%
Use cases	Topology Mapping	14%	4%	20%	33%	7%	50%	42%	7%	69%	61%	16%	85%
	Failure Localization	29%	11%	37%	61%	14%	81%	60%	25%	80%	80%	18%	92%
	Hijack detection	58%	53%	73%	73%	54%	87%	77%	59%	92%	82%	74%	96%

Table 9: Performance of GILL, *Rnd.-VPs* and *best-case* on a simulated mini Internet where the proportion of ASes deploying a VP ranges from 1% to 50%. GILL leverages high coverage but the two other baselines do not.

the triggered events (or existing AS links). In Table 9, for each baseline and each VP deployment scenario, we present the percentage of detected events (or observed links). For GILL, we also present the percentage of retained updates and the percentage of selected anchor VPs, according to each VP deployment scenario. Note that Table 9 shows median results obtained on the *artificial topologies*. We observed that results with the *pruned known AS topology* are similar. We observe the following three takeaways:

Takeway #1: GILL's sampling algorithms are responsive to the VP coverage, effectively coping with a higher VP coverage by discarding more (redundant) data. GILL retains 18% of the updates when coverage is 2%, and 7.9%, 5.4%, and 4.7%, when coverage is 10%, 20%, and 50%, respectively. Similarly, GILL selects 17% of the VPs as anchor VPs when coverage is 2%, and 3.3%, 1.3%, and 0.9% when coverage is 10%, 20%, and 50%, respectively. This behavior is expected, as a higher coverage leads to a higher proportion of redundant updates. Consequently, we expect GILL to be able to effectively scale with a VP deployment that increases by an order of magnitude more VP.

Takeaway #2: GILL 's *overshoot-and-discard* data collection scheme is efficient. While the *best* baseline outperforms GILL, it also collects many more updates. With 50% VP coverage, GILL localizes 80% of p2p links against 92% with *best*. However, GILL collects $\approx$ 21x fewer updates than *best*. When coverage is high (e.g., 50%), the number of updates processed by GILL is comparable to *best*, but with a 2% coverage. Assuming this observation holds in the real Internet, GILL would collect a similar number of updates as RIS and RV today, while peering with 50% of the $\approx$75k ASes, which would triple the number of p2p links observed, double the number of localized failures, and reduce the proportion of undetected hijacks by 33%.

Takeaway #3: GILL outperforms the *current* baseline for all use cases. Even with 50% coverage, only 16% of p2p links are detected (against 61% for GILL) when processing the same number of updates as GILL. The forged-origin hijack use case is the most challenging for GILL, as all prefixes owned by an AS are subject to identical updates in our simulations (thus GILL discards many of these updates) while only one prefix is

hijacked. Nevertheless, GILL consistently outperforms *current* for this use case. For example, GILL detects +18% of hijacks with a 25% coverage.

6.3 Immediate benefits of GILL's sampling

The long-term benefits of GILL will only be visible if it gains traction. However, in this section, we demonstrate that GILL's sampling can improve the accuracy and coverage of two measurement studies. Both these studies sample the data in an unoptimized fashion. The first study, the AS relationship inference, samples data by selecting an arbitrary subset of the VPs. The second study, the characterization of international routing detours, uses a limited time frame of measurement. We demonstrate that, for both of these studies, sampling the data with GILL yields better performances, and can change the conclusion of the analysis. Unlike §6.2, the ground truth is now unknown.

6.3.1 GILL improves AS relationship inference

There exist various methodologies and algorithms to infer the economic relationships between ASes. For this analysis, we focus on the methodology proposed in [35], as the datasets, input data, results, methodology, and source code are publicly available [109]. We compare the number of inferred AS relationships obtained by the algorithm using the original dataset against the same algorithm but using a dataset sampled using GILL (ensuring that both datasets have the same volume). We also show, for each data sampling strategy, the accuracy of the inferences.

Experiment setup. We replicate the methodology proposed in [35] but use GILL to sample the input data. CAIDA has provided the results of this algorithm's inferences every month since 1999. This algorithm takes as input a set of AS paths extracted from one RIB of the first five days of each month. In 2023, it collected data from 648 VPs selected arbitrarily among RIS and RV VPs. CAIDA's website also provides the set of AS paths used to make the inference each month, allowing us to know the volume of data used as input. Therefore, we calibrate GILL's settings to collect the same volume of data as the one used by CAIDA each month of the year 2023. Each month, we collect one RIB of the first five days from **all** the available VPs and then apply GILL's filters, ensuring that both the original dataset and the dataset built by GILL have the same volume. This process ensures a fair comparison between the dataset sampled by GILL and the original dataset, such that any performance gap can be confidently attributed to GILL. We then run the algorithm provided by CAIDA on both the original and GILL's datasets, and compare the number of AS relationships that each dataset enables to infer. Additionally, we compare the *True Positive Rate* (TPR) of each inference. The

TPR corresponds to the percentage of AS relationships that appear in the ground truth and are correctly inferred. We detail how we build this ground truth.

Ground truth construction. The ground truth of AS relationships used for validation is available on the CAIDA website [109]. However, it was updated for the last time in 2013 and since relationships between ASes may be renegotiated, this dataset is likely to contain some outdated validations. Therefore, we build an updated ground truth using the same methodology as described in [35]. It uses two strategies to gather a set of AS relationships, along with a survey conducted among operators, which we did not replicate. The first strategy involves collecting IRR records from every registered AS and parsing the *import* and *export* sections. For two ASes *AS1* and *AS2*, if *AS1* imports *ANY* from *AS2* and *AS2* exports *ANY* to *AS1*, then *AS1* and *AS2* have a c2p relationship. The second strategy also leverages IRR records but parses the *remark* sections. Some ASes describe the different information communities that they are using and document their purposes. Some community values indicate the type of AS relationship between the AS that sets the community and the AS from which the route is received. For instance, Orange sets the community *5511:999* to any route received by a customer. We parse this information for each AS that has an IRR record (when possible) and collect the first RIB of Jan. 2023 from all available VPs. We then combine these two information to extract the RIB entries where one such BGP community appears and use it to retrieve the relationship between the AS that tagged the community and the previous AS in the AS path. These two strategies enable us to gather 58580 distinct AS relationships in our ground truth. We then use this ground truth to evaluate the accuracy of [35] using both the original dataset and the dataset sampled by GILL.

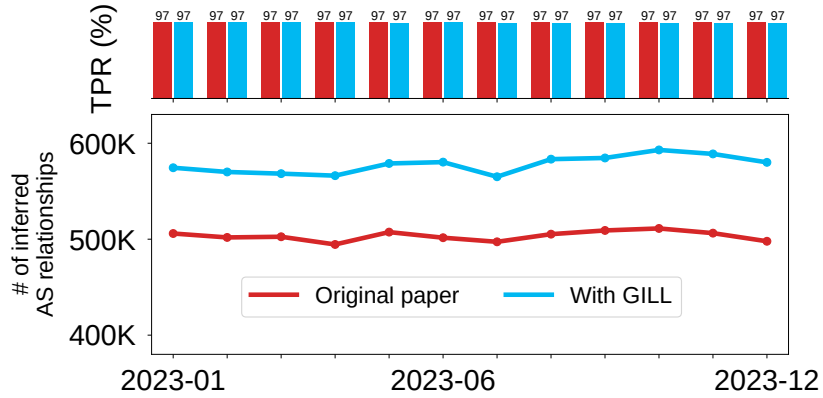


Figure 6.1: Results of the AS relationships inference evaluation. The results in red corresponds to the results of the original dataset while the results in blue corresponds to the results of GILL. GILL enables to infer +16% AS relationships in the median case, while yielding the same inference accuracy.

Experiment results. Fig. 6.1 presents the results of the inferences using both the original dataset and GILL, from Jan. 2023 to Dec. 2023. The graph at the top presents the TPR (in %) of the inferences, while the graph at the bottom presents the number of inferred AS relationships using both strategies. The bottom graph indicates that GILL's sampling enables to consistently observe $\approx$ 89k ($\approx 17\%$) additional AS relationships compared to the original dataset while missing fewer than 8k ($\approx 1.5\%$) of the inferred relationships. Overall, GILL enables to infer consistently $\approx 16\%$ more AS relationships compared to the original dataset. When comparing the accuracy of both inferences, we can see that using either the original dataset or GILL has no impact on the TPR. The TPR consistently remains $\approx 97\%$, regardless of the dataset used or the month at which the inference is made. Therefore, we can conclude that GILL's sampling improves the AS relationship inference and can significantly impact current studies when applied to data collected by RIS and RV.

Impact on AS cone size computation. ASrank [165] is an algorithm that ranks the ASes operating on the Internet by comparing their *Customer Cone Size* (CCS). The CCS is a metric that computes the number of ASes that a given AS can reach using only provider-to-customer links, i.e., those that belong to its customer cone. The customer cone of an *ASX* is composed of *ASX*'s customers as well as *ASX*'s customers' customers and so on. This metric leverages both AS paths and inferred AS relationships to determine which AS can be reached using only p2c links. An AS path is used to compute the CCS of an *ASX* only if the AS path is observed (i.e., collected) by a peer or a provider of *ASX*, mitigating the effects of partial transit relationships between ASes. The algorithm adds to the customer cone of *ASX* the ASes that appear after *ASX* in the AS path if they have either a c2p relationship with *ASX* or a c2p relationship with an AS in *ASX*'s customer cone. Consequently, the CCS is sensitive to both the inferred AS relationships and the diversity of the AS paths used.

We show that GILL's sampling, by processing more diverse AS paths, prevents flawed inferences in the ASrank dataset. We found that the CCS of 1067 changed when using the original dataset or GILL's sampling. We manually investigated a few cases of substantial changes and found that the inferences made using GILL are more accurate. For instance, *AS132337* has an incorrect CCS of 1 in the original dataset (confirmed by *AS132337* itself) and a correct CSS of 18k when using GILL. We observe this inconsistency as GILL selects as an anchor VP the only available VP that collects AS paths with *AS132337* as a transit AS. Similarly, *AS24645* is the route server of Balcan-IX and has an incorrect CSS of 16 in the original ASrank dataset, which is corrected to one when using GILL. These results indicate that GILL enables more accurate inferences of CCSs because it collects more diverse AS paths.

Similarly to the results presented in §6.1, we demonstrated in this section that using GILL to sample BGP data often improves the results of inference algorithms.

6.3.2 GILL improves Routing Detour characterization

We evaluate the impact of GILL's sampling on a study that focuses on characterizing International routing detours [166]. International detours occur when two ASes in the same country are reachable through an AS in another country, potentially leading to extra forwarding delays. We demonstrate that by using fewer updates retained by GILL's sampling, we can extend the duration of the study to identify more detours without processing more data.

Experiment setup. We compare the data collection strategy used in the original paper, which involves collecting all available data during a limited timeframe, against sampling data with GILL over a longer timeframe. The original paper used all the RIBs collected from all VPs during one month (Jan. 2016). We calibrate GILL's sampling to either retain 50% of the data while using all the RIBS for two months or retain 25% of the data while using 4 months of data. Consequently, the number of processed RIB entries remains the same across all three scenarios ($\approx$ 61B), ensuring a fair comparison. To ensure that any performance gap can be confidently attributed to GILL, we also compare the original approach with a random baseline, where we randomly select a subset of the VPs but use data over a longer timeframe. We ensure that the number of processed RIB entries matches the one processed by GILL and the original approach. The random baseline is tested with 30 different random seeds, and we present the results for the median case. We reproduce the original methodology (using publicly available code) on May 2023 and run both GILL's sampling and the random baseline from May 2023 until either June 2023 (when using two months of data) or Aug. 2023 (when using four months of data).

GILL's sampling enables detecting +51% more routing detours. We present the results of the experiment in Table 10. We observe that using a subset of the data sampled with GILL over a longer time frame yields more detected routing detours. Using the original data collection strategy described in [166], we detect 174k routing detours, while using GILL on two months of data enables the detection of 250k routing detours. When using GILL on four months of data (but filtering more RIB entries), we detect even more routing detours, totaling 263k, which corresponds to a +51% improvement in event detection. One might assume that this performance gap is due to the collection strategy of collecting less over a longer time frame. However, when using this strategy with a naive data sampling strategy, the original dataset yields better results. Specifically, when collecting data over four months, the random baseline detects 171k routing detours in the median case, -2% less compared to the original dataset.

Experiment	Duration	# of processed Updates	# of Detours
Original paper	1 Month	≈61B	174k
Random baseline	2 Months	≈61B	165k (median)
	4 Months	≈61B	171k (median)
GILL's sampling	2 Months	≈61B	**250K**
	4 Months	≈61B	**263k**

Table 10: Using the same number of processed updates, using GILL with less retained data on a longer time frame enables detecting more routing detours than using all VPs but on a shorter time frame.

Consequently, we can confidently assert that GILL's sampling enables improving the detection of international routing detours.

GILL's sampling improves the characterization of routing detours. In [166], the authors present a methodology to rank countries and the ASes according to the number of routing detours they experience. We replicate this methodology with the experiment setup described above and show that using GILL's sampling leads to different conclusions compared to when using the original data collection methodology. We find significant differences in the country and AS rankings, including the two following notable cases:
Case I: Using GILL's sampling (with two months of data), we discover 33k (+68%) additional detours traversing the US and 22k (+37%) traversing Russia compared to when using the original settings. These additional detours rank the US as the #1 country with the highest number of routing detours and Russia as #2, whereas with the original settings, Russia is ranked #1 and the US #2.
Case II: Using GILL's sampling (with two months of data) enables detecting 720 (+83%) additional routing detours involving AS262503 compared to when using the original settings. This changes rankings: AS262503 becomes #1 vs. #7 with the original settings. Since our rankings are based on the highest number of routing detours compared to [166], we can confidently assert that GILL improves the characterization of international routing detours.

6.4 GILL is used by existing systems

In this section, we present the impact of GILL's sampling on DFOH [82], a system that detects forged-origin hijacks in the wild. Detecting these forged-origin hijacks requires

processing power, so DFOH must sample BGP data. Since forged-origin hijacks consist of *per-prefix* attacks, a system aiming at detecting them needs a complete view of the prefixes. Consequently, DFOH uses GILL's anchor VPs to sample the data.

We start in §6.4.1 by providing an overview of DFOH, another project I contributed to during my Ph.D. Next, we demonstrate in §6.4.2 the impact of GILL's sampling on DFOH's detection of forged-origin hijacks.

6.4.1 DFOH's workflow

Forged-origin hijacks remain a threat to the Internet even considering a complete deployment of RPKI [74], as the attacker prepends the legitimate origin of the hijacked prefix. This type of attack frequently makes the headlines [80, 81], highlighting the urgent need for a system that detects them. DFOH is a system that detects forged-origin hijacks in the wild. It raises alerts for the entire set of ASes operating on the Internet using only public data as input.

Lack of defenses against forged-origin hijacks. Currently, there exist no mechanism to detect or mitigate forged-origin hijacks in the wild. The two most common methods to mitigate routing attacks are RPKI validation and route filtering. However, none of these two mechanisms are effective in the case of forged-origin hijacks. In fact, as RPKI validation checks the mapping between the announced prefix and the origin AS, it does not enable to filter forged-origin updates.

The second defense mechanism is route filtering, which is encouraged by MANRS [167] to prevent an AS from propagating incorrect routing information. However, in practice, these filters are often missing, inaccurate, or controllable by an attacker. In April 2020, a misconfiguration made by ROSTELECOM resulted in this AS hijacking several prefixes, impacting Akamai and Amazon [168]. It appears that missing filters from Rascom and Cogent led to the propagation of these incorrect announces to a large portion of the Internet. When filters are configured, they often rely on data collected from the IRR [158] to infer the peering information between ASes. Unfortunately, IRRs are known to contain inaccurate information [169, 170, 171]. Controllable filters can occur because data pushed to IRR is often not verified [172]. Specifically, an attacker can pollute the AS-set, an object that specifies to which ASes traffic should be routed. Since filter generation tools use these objects, this can lead to the deployment of illegitimate filters. This is supposedly what happened in 2022 when AS209243 successfully hijacked some Amazon prefixes after adding Amazon's AS in its own AS-set [173].

Current reactive defenses against forged-origin hijacks are narrowly focused on the AS that deploys it. ARTEMIS [69] is a system that monitors prefixes owned by the deploying AS, detecting forged-origin hijacks (among other anomalies) in real time.

The system relies on a key mechanism that classifies new AS links as legitimate when observed in both directions and reports the new links as suspicious otherwise. ARTEMIS therefore exhibits two main limitations. First, it only detects hijacks for the prefixes owned by the AS deploying it, meaning that it cannot detect forged-origin hijacks in the wild. Second, because of the lack of visibility over the routing ecosystem, only a small portion of the AS links are observed in both directions, meaning that relying solely on this feature may result in many false positives (i.e., legitimate links inferred as malicious). This requires more human processing and reduces the value of such alarms. Therefore, DFOH relies on other mechanisms to detect forged-origin hijacks in the wild. DFOH's workflow is illustrated in Fig. 6.2.

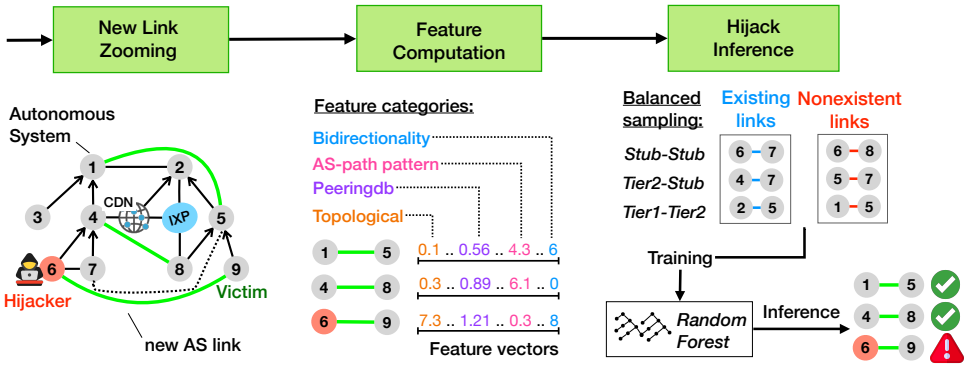


Figure 6.2: DFOH's forged-origin hijack inference pipeline.

Zooming on new AS links. DFOH processes public BGP feeds from RIS and RV (both live and offline) to retrieve AS links that appear for the first time in the last 300 days, to align with parameters found in [69]. DFOH focuses on these new links because forged-origin hijacks are likely to trigger a fake new AS link between the attacker and its victim, as explained in §1.3.4. Conducting a forged-origin hijack without creating a new link (e.g., by prepending an existing path from the attacker to the victim, but violating routing policies) substantially limits the attack surface [82]. However, triggering an alarm whenever a new AS link is detected may result in numerous false alarms being raised, as there are 323 new AS links on average every day (in 2022). Most of the new AS links that are observed are legitimate, resulting from new peering agreements or newly observed backup paths.

DFOH needs to implement a strategy to distinguish fake links from legitimate ones. Unfortunately, as demonstrated in [82], link prediction algorithms such as SEAL [174] performs poorly when applied to the AS-level graph. These algorithms are ineffective because they are generic while detecting malicious routing behaviors requires specific characteristics. Additionally, since the AS-level graph has a hierarchical structure,

generic algorithms are not trained on graphs exhibiting specific structures. Therefore, DFOH uses its own pipeline to detect forged-origin hijacks, starting with the computation of different features carefully selected for the case of BGP.

Feature computation. DFOH must implement features that satisfy the following four requirements: *(i)* the system must be fast (i.e., near real-time), *(ii)* it must be accurate, both in detecting malicious behaviors and avoiding false alarms, *(iii)* it must be robust against missing and polluted data, and *(iv)* it must be efficient in every possible attack scenarios. Therefore, DFOH uses a set of carefully selected features that are classified into the following four categories:

Topological features: These features are the same as those used in §4.2.2 by MVP, which aim at characterizing the topological patterns of a graph. DFOH computes the impact of new AS links on the AS-level topology structure by computing the differences induced by these new AS links on the computed features. A legitimate new AS link is unlikely to disrupt the hierarchical structure of the Internet, while a fake new AS link, because it does not correspond to the typical routing policies, is likely to impact this structure. For each selected topological feature (see Table 3), DFOH computes the difference induced by the new AS link on the feature. For each AS involved in the new link, DFOH computes the topological features *without* the new AS link and *with* the new AS links and then computes the difference between these two values. These features are effective for attack scenarios where an AS with low connectivity hijacks a highly connected AS.

Peering features: These features leverage the peering properties of the two ASes involved in the new AS link, using data gathered from PeeringDB [83] and BGPview [175]. For instance, if the two ASes have a presence in the same IXP or facility, they are more likely to legitimately peer. Similarly, these features also evaluate the geographical characteristics, relying on the intuition that two ASes in the same country are more likely to appear connected in BGP. Obviously, while these characteristics are intuitive, they are not necessarily true; a simple counter-example is remote peering [176, 177]. Fortunately, other feature categories compensate this limitation. This feature category comprises five features computed for each end of the new AS link and described in Table 11:

All these features are computed on the neighbors of the two involved ASes, as an attacker can modify its own peeringDB records to fool the classifier. For each feature, DFOH builds a vector where each component corresponds to one possible element, according to the feature ID in Table 11. On September 19, 2022, for features 1 and 5, the vectors have 271 dimensions, each representing one of the 271 countries found in peeringDB. Similarly, for feature 2, the vectors have 944 (number of IXPs) dimensions,

Index	Description
1	The countries where *ASX*'s neighbors are registered
2	The IXPs to which *ASX*'s neighbors are connected to
3	The facilities to which *ASX*'s neighbors are present
4	The cities of the facilities to which *ASX*'s neighbors are present
5	The countries of the facilities to which *ASX*'s neighbors are present

Table 11: List of peering features used by DFOH along with their description. We consider features computation for *ASX*.

whereas for feature 3 they have 3558 (number of facilities) dimensions, and for feature 4 they have 1482 (number of cities) dimensions. The value of each component of the vector computed for *ASX* corresponds to the number of *ASX*'s neighbors that have a presence in the corresponding country/IXP/facility/city. For each new link, DFOH computes these five vectors for both ASes at each end of the link, one corresponding to each feature presented in Table 11. It then compares them by computing the *cosine distance* between the two vectors computed for both ASes involved in the new link.

AS path features: This feature category uses not only the new link but the entire AS path where the new link appeared. It evaluates how consistent the paths are with typical routing policies. Intuitively, since paths on the Internet are supposed to follow a valley-free pattern, the degrees and the CCS of the ASes in an AS path should follow a similar pattern. However, not all paths on the Internet are valley-free [35], and checking whether a path is valley-free or not (in terms of degree or CCS) might result in inaccurate inferences. Therefore, DFOH trains a classifier that takes a sequence of AS degrees or CCS as an input and returns the probability of this sequence being legitimate. This classifier is trained using both legitimate and illegitimate AS paths. Legitimate paths are actual paths observed in the routing tables, while malicious paths are legitimate paths to which we prepended an additional AS, which corresponds to the victim. We ensure that the link created by the malicious AS path does not exist in the BGP data. The legitimate and malicious synthetic paths are carefully selected, as highlighted later in this workflow description. The final output of these features is a probability (between 0 and 1) of the sequence of AS degree or CCS to be legitimate.

Bidirectionality feature: Similarly to ARTEMIS, DFOH relies on the bidirectionality feature, as a link seen in both directions is likely to be legitimate. To compute this feature, DFOH uses both BGP data to gather AS links and data extracted from IRR. Leveraging IRR data does not enable any adversarial input since the attacker can only

pollute its own IRR records, i.e., the direction from *attacker* to *victim*. Therefore DFOH uses a binary value (0 or 1) to indicate whether the link has been seen in both directions.

Hijack inference. DFOH aggregates all the features described above into vectors, which are used to train a classifier following a supervised training approach used in state-of-the-art link prediction [178, 179]. From the AS-level graph, DFOH samples 60k existent (legitimate) and non-existent (malicious) links that will be used to train a random forest. DFOH uses a grid-search process to estimate the best parameters using 25% of the training dataset, while the remaining 75% is used for the training. Note that the sampled links are different from those selected for the AS-path features.

As DFOH must work regardless of the attack scenario, every possible scenario of new links (legitimate or not) must be represented in the training set. We demonstrated in [82] that a randomly sampled set of links leads to high performances in scenarios where a stub AS hijacks another stub AS but performs poorly for other attack scenarios. Therefore, DFOH first classifies all ASes operating on the Internet into categories by clustering them based on their AS degree and Customer Cone Sizes. It then selects existent and non-existent links, ensuring their distribution matches the distribution of the AS links observed on the Internet, according to their AS categories (computed with the clustering algorithm).

Since the AS path features require an AS path, for each sampled non-existent link, we select an existing AS path where the origin (which will correspond to the attacker) is one end of the sampled link and prepend the other end, which will be the victim. For existent links, we randomly select one AS path where the existent link appears. DFOH then uses all these samples to build a classifier that achieves a 90.9% TPR and a 1.9% FPR. These results mean that hijacked links are classified as malicious in 90.9% of the cases and legitimate links are classified as malicious in only 1.9% of the cases.

6.4.2 Impact of GILL's sampling on DFOH

The computation of some features and the training if the classifier are computationally expensive and DFOH needs to provide hijack detection in real-time. Therefore, DFOH samples data using the anchor VPs computed by GILL, as it requires visibility over all existing prefixes. In this section, we demonstrate the impact of GILL on the detected forged-origin hijacks.

Experiment settings. We developed two modified versions of DFOH by changing the data sampling strategy. DFOH$_R$ uses the same volume as the original version of DFOH but selects the VPs randomly. DFOH$_{ALL}$ uses all the available data and serves

as an approximation of the ground truth (as incorrect inferences are still possible). We limited our analysis to Sept. 2023, as DFOH$_{ALL}$ is computationally expensive.

Experiment results. In Sept. 2023, DFOH$_{ALL}$ infers 1708 new links as suspicious using all available data from RIS and RV. In this paragraph, we define the proportion of cases considered suspicious by DFOH$_{ALL}$ that are detected by the evaluated baseline as the TPR. We refer to the proportion of cases considered suspicious by a baseline, but not by DFOH$_{ALL}$ as FPR. A false positive can occur when a link is considered as new by either DFOH or DFOH$_R$, but not by DFOH$_{ALL}$, since DFOH$_{ALL}$ has a better view over Internet links. The original version of DFOH (which uses GILL anchor VPs) outperforms DFOH$_R$ for both TPR and FPR. DFOH has a TPR of 94.1%, meaning that it detects 94.1% of the suspicious cases, while DFOH$_R$ exhibits a TPR of only 71.5%. DFOH has a FPR of 14.4%, against 60.1% for DFOH$_R$. These results highlight the positive impact of sampling data using GILL when trying to detect forged-origin hijacks.

In this chapter, we demonstrated the effectiveness of GILL's sampling. We show that it outperforms the state-of-the-art in BGP data sampling and that it does not overfit toward any possible objective. We also highlighted both the immediate and long-term benefits of GILL's sampling on well-known measurement studies. Finally, we demonstrated the impact of GILL on DFOH, a system that detects forged-origin hijacks in the wild.

7

Future works

The main focus of this thesis was GILL, a system that improves BGP data collection, which helps both users and BGP data collection platform operators cope with the high volume of data. Complementary approaches exist, aiming at improving the way BGP data is stored and providing tools that facilitate BGP data processing. In this section, we describe two potential follow-up works, focusing on improving BGP data storage and BGP data processing.

In section §7.1, we illustrate the limitations of the current BGP data storage format and propose a new database scheme that enables users to efficiently retrieve precise pieces of the data. In section §7.2, we introduce a new project, Text2BGP, which leverages Large Language Models (LLMs) to provide an interface to the users that facilitates the most common BGP data analysis.

7.1 Improving the collected BGP data format

Currently, public BGP data collection platforms such as RIS and RV store the data using the Multi-Threaded Toolkit (MRT [85]) format. This generic data storage format enables storing routing data related to multiple protocols, including BGP, OSPF, ISIS, or RIP. To parse this data, the community has developed multiple tools aimed at improving

the processing of routing data archives [86, 117, 180, 181, 182]. However, in this section, we show that this format is suboptimal and does not leverage BGP characteristics to reduce the volume of stored data.

Current format. The MRT format has been extensively used to store BGP data for now more than two decades. It consists of a per-line format, where each line contains one unit of routing information. Each MRT entry has a header with three main pieces of information: the timestamp at which the entry was dumped, the entry type/subtype, and the length of the entry payload. Although extended to other routing protocols, we focus here only on the BGP type. There are two main subtypes of MRT files used to store BGP data: one for the RIB and one for the raw BGP updates.

When storing a RIB, each MRT entry represents one RIB entry, meaning there is one entry per prefix in the MRT file. As the BGP RIBS are often dumped per collector, data collected from multiple VPs might be stored in the same MRT file. Therefore, each RIB dump file contains a mapping between all the VPs and a unique ID. In such a mapping, a VP is represented by the ASN deploying it and the IP address of the router hosting the VP. This mapping must be stored at the beginning of the MRT file. Consequently, each RIB entry contains not only the prefix and the BGP attributes but also an ID corresponding to the VP associated with the RIB entry.

When storing raw BGP updates, the format is significantly simpler. Each entry contains the following information: the ASN and the IP of the VP from which the update has been received, the local ASN and IP of the route collector, the address family (IPv4 or IPv6), and the entire BGP update, dumped exactly as it was received. Unlike RIB dumps, the update format does not use any mapping between the peer (VP) and an ID.

Limitation of MRT format. Despite its extensive use, this format exhibits three main limitations, which increase the volume of the dumped files and reduce the practicability for users. The first limitation, though not fundamental, is the per-collector dumps instead of the per-VP dumps. Focusing on a single VPs requires downloading the entire per-collector dump, and searching for the VP within the dump file since there is no easy way to retrieve the information related to a single given VP.

Second, since MRT is a generic format capable of storing information related to multiple protocols, it is not optimized specifically for BGP. The high redundancy in the BGP data presents an opportunity for optimizing the format. As demonstrated in §4.1.1, BGP updates are often recurrent. Therefore, implementing a mapping between BGP attributes and an ID, and storing only the ID in the MRT entry each time an update appears could significantly reduce the volume of the files. Additionally, since many prefixes share the exact same attributes, there is another opportunity to save space in the dumps by grouping these similar prefixes.

Third, MRT consists of a per-line storage format. However, users are often interested in a specific portion of the data. For example, an operator may want to monitor only the prefixes belonging to its AS. Retrieving data about specific prefixes requires processing the entire file, as MRT does not implement a way to focus only on a subset of any attributes. Answering the question "*What was the AS path of prefix p at time t on VP v?*" requires processing the entire last RIB dump file before time t to find prefix p on VP v and then processing every single update until time t. We find that this process takes on average 50 seconds using BGPstream on RIS and RV data. We tested this 100 times each time varying the VP, prefix, and time randomly to mitigate the bias induced by the differences in the size of the MRT files. Therefore, we can confidently assert that per-line storage does not optimize such search query in the case of BGP data.

New data storage strategy: per-column storage. We explore a per-column data storage scheme for BGP data. To conduct this experiment, we rely on generic per-column storage databases: PostgreSQL [183]. Note that while there exist other per-column databases, we choose PostgreSQL due to its extensive documentation. For each VP, we define two types of tables, the RIB tables and the update table.

Since RIB dumps occur every two hours for RV and eight hours for RIS, we need to have multiple RIB tables, one for each distinct dump. Each RIB table contains three columns, one for the prefix, one for the AS path, and one for the BGP communities. Note that no information about the time or the VP is stored, as this information is embedded in the table name. While we choose to store only the two main BGP attributes, this table scheme can be adapted to include other attributes, such as the large communities, without fundamentally changing how the database works in practice.

All the BGP updates received from a given VP are stored in the same table, as BGP updates do not consist of regular snapshots. Each row in the table represents one BGP update. To retrieve the timestamp at which the update has been received, an additional column corresponding to the timestamp is included. Similar to the RIB tables, no information about the VP that received the route is stored, as it is embedded in the table name.

This new storage implements per-column storage that enables users to efficiently retrieve all entries matching a given property (e.g., prefix or AS path). If an operator wants to get all updates received for one of its prefixes, a single lookup on the prefix column enables focusing only on the corresponding rows. With this new database scheme, answering the question "*What was the AS path of prefix p at time t on VP v?*" requires 0.095 seconds on average using the same parameters as described above, yielding a significant improvement over the current state of the art for BGP data storage. Despite being more efficient at answering specific queries quickly, this database scheme

does not leverage BGP properties to optimize the volume of data stored. Therefore, we propose a further improvement of the database scheme, optimizing both queries performance and volume of data stored for the case of BGP.

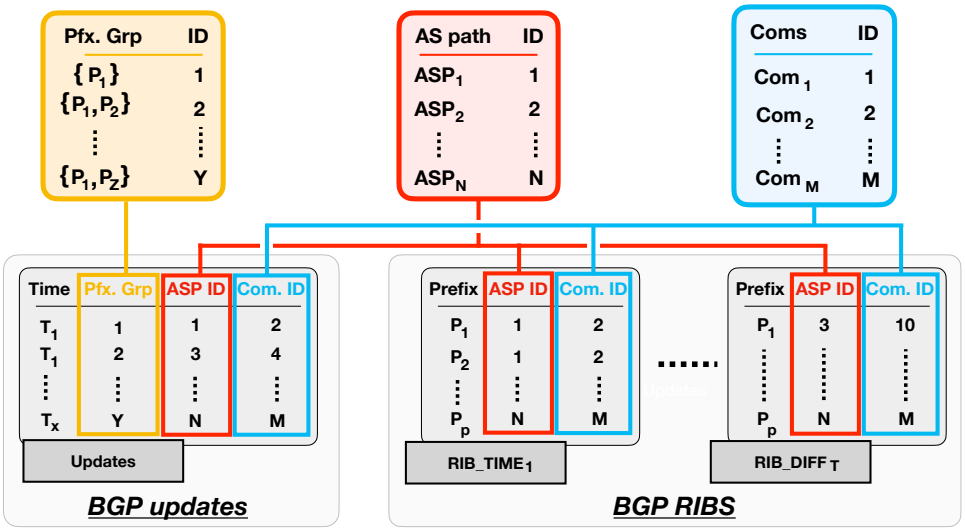


Figure 7.1: Optimized database scheme. The proposed tables enable fast search queries, while optimizing the data volume for the case of BGP. This figure represents the tables used for one single VP.

Fine-tuning storage for BGP data. In this paragraph, we enhance the previously described database scheme to optimize both search queries and data volume. We propose general optimizations applicable to both RIB and update tables, as well as specific optimizations for each table type.Fig. 7.1 presents the proposed database scheme used for a single VP.

General optimization: As explained in §3, some BGP attributes appear regularly in BGP data. Therefore, we rely on mapping tables where each BGP attribute is associated with a unique ID. For instance, in Fig. 7.1, there is a table (red) that associates every observed AS path with a unique ID. Similarly, each set of BGP community (blue table) is associated with a unique ID. Since an ID is stored in four bytes and both AS path and BGP communities often require more than four bytes to be stored, storing only the ID of the BGP attributes significantly optimizes the volume of data stored. Although a search query implements now an additional level of indirection, comparing an ID is faster than comparing strings (e.g., AS path or community values). Consequently, this indirection does not significantly increase the time required to answer a search query. Note that we represented only the AS paths and the community values as they are the

most useful attributes. However, this database scheme can be extended to include more attributes.

Optimization for Update tables: As described in §3, prefixes announced by the same AS are likely to share the exact same BGP attributes [184]. Therefore, instead of having duplicated lines for each of these prefixes, we implement a column *"Pfx. grp"* that stores a list of prefixes. Additionally, since these groups of prefixes are likely to appear recurrently, we store only an ID that corresponds to the set of prefixes. This mapping is stored in a table that associates a set of prefixes to a unique ID (yellow table on Fig. 7.1).

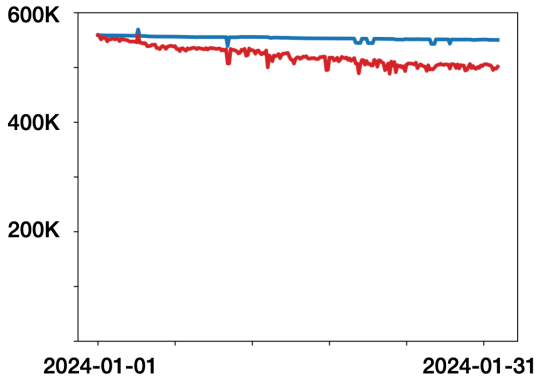


Figure 7.2: Average number of identical RIB entries according to the time between the two dumps.

Optimization for RIB tables: RV stores a RIB dump every two hours. However, a significant portion of the entries is likely to be identical between two consecutive dumps. We confirm this intuition by collecting the first RIBs from route-views3 of March 2024. We then collect all other RIBs in March 2024 and measure the proportion of entries that are identical (using a per-prefix comparison) to the first RIB. Fig. 7.2 presents the average number of identical entries between two RIB dumps (y-axis) according to the time between these two dumps (x-axis). We can observe that even after one month, the percentage of identical entries remains high on average (≈87%). Therefore, we store only one RIB at the beginning of each month and then store only the difference between the subsequent RIB dumps and the first one. We chose one month as a threshold as we experimentally observed that the number of identical entries significantly decreased after 30 days.

The proposed database scheme optimizes both the search queries and the data storage. In March 2024, all the MRT files with Bzip2 compression combined (i.e., both RIB and updates) accounted for 37GB for collector route-views3. Our database scheme accounted for 38GB of data without any other compression and enables significantly

faster search queries. Similar work exists, such as BGP2GO, a tool that archives all the MRT files from RouteViews [161]

7.2 Enhancing BGP data processing.

To process BGP data, users often rely on BGP data collection software such as BGPstream [86] or BGPkit [117]. While these tools facilitate data collection, users still need to generate the code required to conduct their specific analysis. Although some studies require very specific information, most require more generic information that is common to multiple measurement studies. However, there is no straightforward way for users to retrieve this data, resulting in researchers often writing the same portion of code multiple times. Therefore, a tool that automates BGP data analysis could help researchers to save time and resources.

Opportunity with LLM. Large Language Models are Machine Learning tools that can process Natural Language (NL) to answer questions [185], generate images [186], or generate code [187]. There also exist other use cases where LLM can be applied, including network-related fields [188, 189, 190]. However, current LLMs cannot use BGP data collected from RIS and RV to answer routing-related questions. For example, the most popular LLM, chatGPT [185], returns the following answers upon asking BGP-related question: *"I'm sorry, but I don't have access to real-time or historical records of network routing information"*. Therefore, we propose fine-tuning LLM to access BGP data and answer recurrent BGP routing questions, saving time for both researchers and network operators.

Gathering a set of recurrent questions. Identifying the most recurrent BGP-related questions asked by both network operators and researchers is challenging as there exists no dataset containing this information. Therefore, we developed a tool to extract all the BGP-related questions from both research papers and network operators' blog posts. This tool works in two steps:
Step #1: Our tool implements a scrapper that extracts text from top conferences in computer networks (SIGCOMM, NSDI, and IMC) and popular network operators' public mailing lists (NANOG). When extracting information from the conference papers, our tool retrieves all materials associated with the papers, such as the abstract and the URL of the PDF. It then pulls the PDF and extracts the text from it. Currently, our tool does not enable processing images or tables inside the PDFs. For the mailing lists, our tool extracts only the main body of each email, as any other information (e.g., the email address of the sender or the date on which the email has been sent) is not useful for extracting BGP-related questions. This work does not raise any ethical issues, as we only use data publicly available and we do not disclose any personal information about the authors (e.g., name of email addresses).

Step #2: Our tool uses LLMs (specifically chatGPT, as we found it to be the most efficient) to extract all the BGP-related questions from the text gathered in step #1. The questions are extracted by asking the following prompt to the LLM: *"Given the following text, extract all questions related to BGP data. If there is no such question in the text, return an empty answer"*. We manually validated this approach by checking 5% of the extracted text and found that the LLM has a TPR of 88% and a FPR of 5%. The TPR corresponds to the proportion of texts where both we and the LLM found a BGP-related question. The FPR corresponds to the proportion of texts where we manually found a BGP-related question but the LLM did not. At this point, our tool identified 117 different BGP-related questions.

After this step, we manually grouped questions with similar keywords, as they require similar data processing. For instance, the question *"What is the list of the transit ASes"* and *"Give me ten transit ASes"* are grouped together. After this manual investigation, we consolidated the questions into 56 distinct ones. These questions are now used to fine-tune the LLM. We will now detail our plan to train an LLM that enables automatic processing of BGP data.

Code generation with LLM. There exist various strategies to enable an LLM to automatically analyze data it does not have access to. For instance, some researchers proposed RestGPT, a tool [191] that leverages LLM to transform Natural Language (NL) into API requests and execute these requests. For instance, this tool allows a user to ask *"play the next song"* using NL, and the tool transforms it into an API request that will play the next song. Other strategies involve transforming NL into SQL queries [192], executing them, and retrieving the results.

Unfortunately, the first strategy overfits toward the capabilities enabled by the API, resulting in poor performances for future analysis that users may want to conduct but are not supported by the API. The second strategy, on the contrary, is too general and can handle only very simple questions. For example, a user asking *What are the ten transit ASes with the largest transit degree* cannot be answered with a simple SQL query. Therefore, we rely on code generation to provide a good tradeoff between practicality and the ability to answer future requests.

Generating code to answer any of the 56 BGP-related questions identified in the previous section is a challenging task, as some questions require very specific portions of code. For example, pulling data from the database is not straightforward for the LLM, as it requires knowing the exact database scheme and how each component interacts with each other. Therefore, we rely on function specifications that are added to the LLM

prompt. Instead of generating code for very specific tasks (e.g., database lookup), the LLM can directly use the functions described in the specification.

Answering questions from users. While the 56 extracted BGP-related questions are representative of the most recurrent questions asked by researchers and network operators, there can be multiple different formulations of the same question. For each of the 56 questions, we asked chatGPT to provide 20 different formulations, which we manually validated. We then fine-tune the LLM (GPT 3.5 in our case) with the 56*20 questions and their corresponding ground truth code. The ground truth consists of manually generated code (using the functions described in the specification) that answers each of the 56 BGP-related questions.

While we already designed the pipeline, we still need to implement and test it. In addition, this strategy can be improved by enhancing the prompt given to the LLM. For instance, by giving some ground truth examples to the LLM, we hope that it will maximize the correctness of the answers. We also plan to survey network operators to know precisely which questions they are likely to ask to the model, and use these questions as a testing set. We are also working on systematic strategies to improve the function specification given as input.

While this work is still preliminary, we believe there is potential for LLM to automate and facilitate BGP data processing.

8

Conclusion

In this thesis, we demonstrated the lack of visibility of current BGP data collection platforms and illustrated how this visibility gap can negatively impact common measurement studies. A naive solution would be to increase the VP coverage; however impossible since both BGP data collection platforms and users already face data management problems, even with the current (low) VP coverage. We identified the opportunity of leveraging the high level of redundancy in the BGP data to reduce the volume of data to process while minimizing the information loss.

The main contribution of this thesis is GILL, a system that uses two carefully selected definitions of redundancy to build a set of filters that retain only the most useful portion of the data and provide effective sampling regardless of the user's objective. GILL relies on two algorithms to evaluate the redundancy in a general fashion: *(i)*, BUS, an algorithm that uses a new metric, the reconstitution power, to classify the BGP updates as redundant or non-redundant and *(ii)*, MVP, an algorithm that evaluates and compare the VPs by computing the impact induced by BGP events on the partial view of each VP. The final result of this work is a new BGP data collection platform implementing GILL's BGP data sampling strategy, which is currently up and running. This platform retains and publishes only the most useful BGP data from multiple ASes on the Internet. GILL's system is bootstrapped with all peering sessions from RIS and RV, providing a head start of visibility. The redundancy evaluation algorithms BUS

and MVP are presented in §4.1 and§4.2, while the GILL system and implementation are presented in §5. In this thesis, we made the following five contributions:

- We estimated the gap of visibility induced by the current low VP coverage using simulations on a mini Internet. We evaluated our ability to effectively conduct three different measurement studies with the current VP coverage.
- We provided a comprehensive analysis of redundancy in the BGP data, using three gradually stricter definitions of redundancy.
- We demonstrated the importance of finding a general definition of redundancy that enables effective sampling regardless of the user's objective. We then provided two definitions of redundancy, one that evaluates the redundancy at the update granularity and the other at the VP granularity.
- We proposed GILL a system that implements an overshoot-and-discard strategy to retain only the most useful information from the BGP data. We implemented and published a BGP data collection platform that uses GILL's algorithms.
- We extensively evaluated our system, the quality of the sampling, and the soundness of the key design choices. We reproduced well-known studies, demonstrating that GILL's sampling improve their accuracy.

In this thesis, we tackled the technical challenges induced by a drastic expansion of the BGP data collection platforms. As explained in §6.2, GILL will only be efficient if it gains traction, which requires motivating operators to establish a peering session with GILL. While this challenge could be the subject of an entirely new contribution, we provide here two possible incentives:

Custom services that improve visibility . In return for peering, GILL could let the network operator configure forwarding rules such that GILL forwards some updates to the operator's network prior to discarding them. Forwarding rules would typically enable operators to have high visibility of their prefixes. If GILL had 100% coverage of VPs, operators could make hijack detection systems such as ARTEMIS [69] bulletproof for their prefixes.

Collective action. Recent community-driven routing security initiatives such as MANRS [167] or VIPzone [193] could encourage participants to contribute BGP data to public BGP data collection platforms. The FCC's recent notice of proposed regulation [194] to require disclosure of BGP security strategies could lend further motivation to share BGP data.

We hope that GILL will gain traction, as having a more complete view of the Internet routing ecosystem can improve the accuracy of BGP measurement and expand the horizons of our understanding of the Internet routing ecosystem.

References

[1] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, IETF, 2006.

[2] Bleeping Computer. Facebook, instagram, and whatsapp back online after bgp fix, 2021. https://www.bleepingcomputer.com/news/technology/facebook-instagram-and-whatsapp-back-online-after-bgp-fix/.

[3] RIPE. RIPE RIS Raw Data, 1999. https://www.ripe.net/data-tools/stats/ris/.

[4] Oregon Univ. Route Views Project, 2001. www.routeviews.org/.

[5] Alexandros Milolidakis, Tobias Bühler, Kunyu Wang, Marco Chiesa, Laurent Vanbever, and Stefano Vissicchio. On the effectiveness of bgp hijackers that evade public route collectors. In *IEEE Access*, 2023.

[6] Ying Zhang, Zheng Zhang, Z. Morley Mao, Y. Charlie Hu, and Bruce M. Maggs. On the impact of route monitor selection. In *ACM IMC*, 2007.

[7] Luca Cittadini, Stefano Vissicchio, and Benoit Donnet. On the quality of BGP route collectors for iBGP policy inference. In *IFIP '14*, 2014.

[8] Enrico Gregori, Alessandro Improta, Luciano Lenzini, Lorenzo Rossi, and Luca Sani. On the Incompleteness of the AS-Level Graph: A Novel Methodology for BGP Route Collector Placement. In *ACM IMC*, 2012.

[9] Zhuoqing Morley Mao, Jennifer Rexford, Jia Wang, and Randy H Katz. Towards an accurate as-level traceroute tool. In *ACM SIGCOMM*, 2003.

[10] BT Group Connected Earth Online Museum. The telegraphic age daws. In *Lexington Books*, 2010.

[11] Statista. Internet of things (iot) and non-iot active device connections worldwide from 2010 to 2025, 2024. URL https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/.

[12] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1959.

[13] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 1958.

[14] John T. Moy. OSPF version 2. RFC 2328, IETF, 1998.

[15] R. Callon. Use of osi is-is for routing in tcp/ip and dual environments. RFC 1195, IETF, 1990.

[16] CIDR. CIDR REPORT, 2023. https://www.cidr-report.org/as2.0/.

[17] Cengiz Alaettinoglu. Scalable router configuration for the internet. *IEEE IC3N*, 1996.

[18] Huston Goeff. Interconnection, peering, and settlements. *Proc. INET*, 1999.

[19] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM ToN*, 2001.

[20] Thomas Krenc, Matthew Luckie, Alexander Marder, and kc claffy. Coarse-grained inference of bgp community intent. In *ACM IMC*, 2023.

[21] Thomas Krenc, Robert Beverly, and Georgios Smaragdakis. As-level bgp community usage classification. In *ACM IMC*, 2021.

[22] Thomas Krenc, Robert Beverly, and Georgios Smaragdakis. Keep your communities clean: exploring the routing message impact of bgp communities. In *ACM CoNEXT*, 2020.

[23] Benoit Donnet and Olivier Bonaventure. On bgp communities. *ACM SIGCOMM CCR*, 2008.

[24] Matthew Roughan, Simon Jonathan Tuke, and Olaf Maennel. Bigfoot, sasquatch, the yeti and other missing links: what we don't know about the as graph. In *IMC*. ACM, 2008.

[25] Hyunseok Chang, Ramesh Govindan, Sugih Jamin, Scott J. Shenker, and Walter Willinger. Towards capturing representative as-level internet topologies. *SIGMETRICS*, 2002.

[26] Wolfgang Mühlbauer, Anja Feldmann, Olaf Maennel, Matthew Roughan, and Steve Uhlig. Building an as-topology model that captures route diversity. *SIGCOMM CCR*, 2006.

[27] Beichuan Zhang, Raymond Liu, Daniel Massey, and Lixia Zhang. Collecting the internet as-level topology. *SIGCOMM CCR*, 2005.

[28] Yihua He, Georgos Siganos, Michalis Faloutsos, and Srikanth Krishnamurthy. A systematic framework for unearthing the missing links: Measurements and impact. In *NSDI*. USENIX, 2007.

[29] R. Cohen and D. Raz. The internet dark matter - on the missing links in the as connectivity map. In *IEEE INFOCOM*, 2006.

[30] Bernhard Ager, Nikolaos Chatzis, Anja Feldmann, Nadi Sarrar, Steve Uhlig, and Walter Willinger. Anatomy of a large european ixp. In *SIGCOMM*. ACM, 2012.

[31] Benoit Donnet and Timur Friedman. Internet topology discovery: a survey. *IEEE CST*, 2007.

[32] Lixin Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Transactions on Networking*, 2001.

[33] L. Subramanian, S. Agarwal, J. Rexford, and R.H. Katz. Characterizing the internet hierarchy from multiple vantage points. In *IEEE INFOCOM*, 2002.

[34] Xenofontas Dimitropoulos, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Young Hyun, kc claffy, and George Riley. As relationships: inference and validation. *SIGCOMM CCR*, 2007.

[35] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and kc claffy. As relationships, customer cones, and validation. In *IMC*. ACM, 2013.

[36] Bruno Quoitin and Olivier Bonaventure. A survey of the utilization of the bgp community attribute. *IETF*, 2002.

[37] Ricardo Oliveira, Dan Pei, Walter Willinger, Beichuan Zhang, and Lixia Zhang. The (in)completeness of the observed internet as-level structure. *IEEE/ACM ToN*, 2010.

[38] Enrico Gregori, Alessandro Improta, Luciano Lenzini, Lorenzo Rossi, and Luca Sani. BGP and Inter-AS Economic Relationships. In *IFIP*, 2011.

[39] Jianhong Xia and Lixin Gao. On the evaluation of as relationship inferences [internet reachability/traffic flow applications]. In *IEEE GLOBECOM*, 2004.

[40] G. Di Battista, M. Patrignani, and M. Pizzonia. Computing the types of the relationships between autonomous systems. In *IEEE INFOCOM 2003*, 2003.

[41] Yuchen Jin, Colin Scott, Amogh Dhamdhere, Vasileios Giotsas, Arvind Krishnamurthy, and Scott Shenker. Stable and practical as relationship inference with problink. In *USENIX NSDI*, 2019.

[42] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed internet routing convergence. In *SIGCOMM*. ACM, 2000.

[43] Timothy G Griffin and Brian J Premore. An experimental analysis of bgp convergence time. In *ICNP*. IEEE, 2001.

[44] Timothy G Griffin and Gordon Wilfong. An analysis of bgp convergence properties. *ACM SIGCOMM CCR*, 1999.

[45] Nick Feamster, David G. Andersen, Hari Balakrishnan, and M. Frans Kaashoek. Measuring the effects of internet path faults on reactive routing. *SIGMETRICS*, 2003.

[46] Zhuoqing Mao, Randy Bush, Timothy Griffin, and Matthew Roughan. Bgp beacons. In *IMC*. ACM, 2003.

[47] Nate Kushman, Srikanth Kandula, Dina Katabi, and Bruce M Maggs. R-bgp: Staying connected in a connected world. *NSDI*, 2007.

[48] Thomas Holterbach, Edgar Costa Molero, Maria Apostolaki, Alberto Dainotti, Stefano Vissicchio, and Laurent Vanbever. Blink: Fast connectivity recovery entirely in the data plane. In *USENIX NSDI*, 2019.

[49] Dan Pei, Matt Azuma, Dan Massey, and Lixia Zhang. Bgp-rcn: Improving bgp convergence through root cause notification. *Computer Networks*, 2005.

[50] Thomas Holterbach, Stefano Vissicchio, Alberto Dainotti, and Laurent Vanbever. Swift: Predictive fast reroute. In *ACM SIGCOMM*, 2017.

[51] Mohit Lad, Xiaoliang Zhao, Beichuan Zhang, Daniel Massey, and Lixia Zhang. Analysis of bgp update surge during slammer worm attack. In *Lecture Notes in CS*, 2003.

[52] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. Inside the slammer worm. *IEEE S&P*, 2003.

[53] Freedman Avi. Isp security talk. In *NANOG*, 2003.

[54] Cliff Changchun Zou, Weibo Gong, and Don Towsley. Code red worm propagation modeling and analysis. In *ACM conference on Computer and communications security*, 2002.

[55] David Moore, Colleen Shannon, and K Claffy. Code-red: a case study on the spread and victims of an internet worm. In *ACM SIGCOMM Workshop on Internet measurment*, 2002.

[56] James Cowie, A Ogielski, BJ Premore, and Yougu Yuan. Global routing instabilities triggered by code red ii and nimda worm attacks. Technical report, Tech. Rep., Renesys Corporation, 2001.

[57] A Machie, Jenssen Roculan, Ryan Russell, and MV Velzen. Nimda worm analysis. Technical report, Tech. Rep., Incident Analysis, SecurityFocus, 2001.

[58] Kenjiro Cho, Cristel Pelsser, Randy Bush, and Youngjoon Won. The japan earthquake: the impact on traffic and routing observed by a local isp. In *SWID*. ACM, 2011.

[59] Carisimo Esteban, Kumar Rashna, J. Wang Caleb, Klein Santiago, and E. Bustamante Fabian. Ten years of the venezuelan crisis - an internet perspective. In *ACM SIGCOMM*, 2024.

[60] Anja Feldmann, Oliver Gasser, Franziska Lichtblau, Enric Pujol, Ingmar Poese, Christoph Dietzel, Daniel Wagner, Matthias Wichtlhuber, Juan Tapiador, Narseo Vallina-Rodriguez, et al. The lockdown effect: Implications of the covid-19 pandemic on internet traffic. In *ACM IMC*, 2020.

[61] Mattijs Jonker, Gautam Akiwate, Antonia Affinito, kc Claffy, Alessio Botta, Geoffrey M Voelker, Roland van Rijswijk-Deij, and Stefan Savage. Where. ru? assessing the impact of conflict on russian domain infrastructure. In *ACM IMC*, 2022.

[62] RIPE. Youtube hijacking: A ripe ncc ris case study, 2008.

[63] MANRS. MANRS blogpost: BGP security in 2021, 2021. https://www.manrs.org/2022/02/bgp-security-in-2021/.

[64] Global Routing Intelligence Platform, 2019. https://grip.inetintel.cc.gatech.edu/.

[65] Johann Schlamp, Ralph Holz, Quentin Jacquemart, Georg Carle, and Ernst W Biersack. Heap: reliable assessment of bgp hijacking attacks. *IEEE Journal on Selected Areas in Communications*, 2016.

[66] Jian Qiu, Lixin Gao, Supranamaya Ranjan, and Antonio Nucci. Detecting bogus bgp route information: Going beyond prefix hijacking. In *SecureComm*, 2007.

[67] Xin Hu and Z Morley Mao. Accurate real-time identification of ip prefix hijacking. In *S&P*. IEEE, 2007.

[68] Lancheng Qin, Dan Li, Ruifeng Li, and Kang Wang. Themis: Accelerating the detection of route origin hijacking by distinguishing legitimate and illegitimate MOAS. In *USENIX Security*, 2022.

[69] Pavlos Sermpezis, Vasileios Kotronis, Petros Gigis, Xenofontas Dimitropoulos, Danilo Cicalese, Alistair King, and Alberto Dainotti. Artemis: Neutralizing bgp hijacking within a minute. *IEEE/ACM ToN*, 2018.

[70] Massimo Candela and al. Bgpalerter, 2019. https://labs.ripe.net/author/massimo_candela/easy-bgp-monitoring-with-bgpalerter/.

[71] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *SP'17*, 2017.

[72] Cecilia Testart, Philipp Richter, Alistair King, Alberto Dainotti, and David Clark. Profiling bgp serial hijackers: capturing persistent misbehavior in the global routing table. In *ACM IMC*, 2019.

[73] Shinyoung Cho, Romain Fontugne, Kenjiro Cho, Alberto Dainotti, and Phillipa Gill. Bgp hijacking classification. In *TMA*. IEEE, 2019.

[74] R. Bush and R. Austein. The resource public key infrastructure (rpki) to router protocol. RFC 6810, IETF, 2013.

[75] Taejoong Chung, Emile Aben, Tim Bruijnzeels, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, Roland van Rijswijk-Deij, John Rula, et al. Rpki is coming of age: A longitudinal study of rpki deployment and invalid route origins. In *ACM IMC*, 2019.

[76] Yossi Gilad, Avichai Cohen, Amir Herzberg, Michael Schapira, and Haya Shulman. Are we there yet? on rpki's deployment and security. *Cryptology Archive*, 2016.

[77] Cecilia Testart, Philipp Richter, Alistair King, Alberto Dainotti, and David Clark. To filter or not to filter: Measuring the benefits of registering in the rpki today. In *PAM*, 2020.

[78] Andreas Reuter, Randy Bush, Italo Cunha, Ethan Katz-Bassett, Thomas C Schmidt, and Matthias Wählisch. Towards a rigorous methodology for measuring adoption of rpki route validation and filtering. *SIGCOMM CCR*, 2018.

[79] Weitong Li, Zhexiao Lin, Mohammad Ishtiaq Ashiq Khan, Emile Aben, Romain Fontugne, Amreesh Phokeer, and Taejoong Chung. RoVista: Measuring and Understanding the Route Origin Validation (ROV) in RPKI. In *ACM IMC*, 2023.

[80] Aftab Siddiqui. KlaySwap – Another BGP Hijack Targeting Crypto Wallets, 2022. https://www.manrs.org/2022/02/klayswap-another-bgp-hijack-targeting-crypto-wallets/.

[81] Celer Bridge incident analysis, 2022. https://www.coinbase.com/blog/celer-bridge-incident-analysis.

[82] Thomas Holterbach, Thomas Alfroy, Amreesh Phokeer, Alberto Dainotti, and Cristel Pelsser. A System to Detect Forged-Origin BGP Hijacks. In *NSDI'24*, 2023.

[83] PeeringDB. The Interconnection Database, 2010. https://www.peeringdb.com/.

[84] Bruno Quoitin, Cristel Pelsser, Louis Swinnen, Olivier Bonaventure, and Steve Uhlig. Interdomain traffic engineering with bgp. *IEEE Comm. mag.*, 2003.

[85] Larry Blunk, Craig Labovitz, and Manish Karir. Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format. In *RFC 6396*, 2011.

[86] Chiara Orsini, Alistair King, Danilo Giordano, Vasileios Giotsas, and Alberto Dainotti. BGPStream: A Software Framework for Live and Historical BGP Data Analysis. In *ACM IMC*, 2016.

[87] Lorenzo Ariemma, Mariano Scazzariello, and Tommaso Caiazzi. MRT#: a Fast Multi-Threaded MRT Parser. In *IFIP/IEEE IM*, 2021.

[88] RIPE. RIPE RIS Peers list, 2023. https://www.ris.ripe.net/peerlist/.

[89] University of Oregon. Route Views Peers list, 2023. http://www.routeviews.org/peers/peering-status.html.

[90] Yihua He, Michalis Faloutsos, Srikanth Krishnamurthy, and Bradley Huffaker. On routing asymmetry in the internet. In *GLOBECOM*. IEEE, 2005.

[91] Brice Augustin, Balachander Krishnamurthy, and Walter Willinger. Ixps: mapped? In *ACM IMC*, 2009.

[92] Alexander Marder, Matthew Luckie, Amogh Dhamdhere, Bradley Huffaker, KC Claffy, and Jonathan M Smith. Pushing the boundaries with bdrmapit: Mapping router ownership at internet scale. In *ACM IMC*, 2018.

[93] Kai Chen, David R Choffnes, Rahul Potharaju, Yan Chen, Fabian E Bustamante, Dan Pei, and Yao Zhao. Where the sidewalk ends: Extending the internet as graph using traceroutes from p2p users. In *ACM CoNEXT*, 2009.

[94] Esteban Carisimo, Julián M Del Fiore, Diego Dujovne, Cristel Pelsser, and J Ignacio Alvarez-Hamelin. A first look at the latin american ixps. *ACM SIGCOMM CCR*, 2020.

[95] BGPWatch. Bgpwatch: Bgp routing analysis and diagnostic platform, 2023. https://bgpwatch.cgtf.net/.

[96] Alessandro Improta. Isolario project: The real-time internet routing observatory, 2023. https://content.cooperate.com/post/internet_history/.

[97] Kentik. Introducing bgp monitoring from kentik, 2022. https://www.kentik.com/blog/bgp-monitoring-from-kentik/.

[98] Cisco. Bgpmon, 2019. https://www.bgpmon.net/services/route-monitoring/.

[99] ThousandEyes. Thousandeyes bgp monitors, 2024. https://medium.com/thousandeyes-engineering/thousandeyes-bgp-monitors-fac4d272609c.

[100] Thousand Eyes. Monitor bgp routes to and from your network, 2024. https://www.thousandeyes.com/solutions/bgp-and-route-monitoring.

[101] Radar Qrator. Real-time bgp monitoring, 2024. https://radar.qrator.net/solution.

[102] PacketVis. Packetvis, 2022. https://packetvis.com/,.

[103] Ben Cox. BGP tools, 2023. https://bgp.tools/.

[104] Daniel Walton, Alvaro Retana, Enke Chen, and John Scudder. Advertisement of Multiple Paths in BGP. RFC 7911, 2016.

[105] John Scudder, Rex Fernando, and Stephen Stuart. BGP Monitoring Protocol (BMP). RFC 7854, 2016.

[106] Alessandro Improta. some thoughts on add-path and bmp at ripe/ris, 2023. https://www.ripe.net/ripe/mail/archives/mat-wg/2023-March/001015.html.

[107] B. Quoitin and S. Uhlig. Modeling the routing of an autonomous system with c-bgp. *IEEE Network*, 2005.

[108] Murtaza Motiwala, Megan Elmore, Nick Feamster, and Santosh Vempala. Path splicing. In *SIGCOMM '08*. ACM, 2008.

[109] University San Diego. The CAIDA AS Relationships Dataset, 2022, 2022. https://www.caida.org/catalog/datasets/as-relationships/.

[110] Rodrigo Aldecoa, Chiara Orsini, and Dmitri Krioukov. Hyperbolic graph generator. In *Computer Physics Communications*, 2015.

[111] Anja Feldmann, Olaf Maennel, Z. Morley Mao, Arthur Berger, and Bruce Maggs. Locating internet routing instabilities. *ACM SIGCOMM*, page 205–218, 2004.

[112] Nikolaos Chatzis and al. On the benefits of using a large ixp as an internet vantage point. In *ACM IMC*, 2013.

[113] Ahmed Elmokashfi and Amogh Dhamdhere. Revisiting bgp churn growth. *ACM SIGCOMM CCR*, 2013.

[114] Emile Aben. Route Collection at the RIPE NCC - Where are we and where should we go?, 2020. https://labs.ripe.net/author/emileaben/.

[115] Emile Aben. Two years of selective peering with ris, 2023. https://labs.ripe.net/author/emileaben/two-years-of-selective-peering-with-ris/.

[116] Robert Kisteleki. RIPE NCC Measurement Data Retention Principles, 2023. https://labs.ripe.net/author/kistel/ripe-ncc-measurement-data-retention-principles/.

[117] BGPKIT. BGPKIT, 2022. https://blog.bgpkit.com/.

[118] Pavlos Sermpezis, Lars Prehn, Sofia Kostoglou, Marcel Flores, Athena Vakali, and Emile Aben. Bias in Internet Measurement Platforms. In *TMA'23*, 2023.

[119] Timm Böttger, Félix Cuadrado, and Steve Uhlig. Looking for hypergiants in PeeringDB. In *SIGCOMM CCR*, 2018.

[120] Lars Prehn and Anja Feldmann. How Biased is Our Validation (Data) for AS Relationships? In *ACM IMC*, 2021.

[121] Mattia Tantardini, Francesca Ieva, Lucia Tajoli, and Carlo Piccardi. Comparing methods for comparing networks. In *Nature*, 2019.

[122] Paolo Boldi and Sebastiano Vigna. Axioms for centrality, 2013.

[123] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1978.

[124] Jari Saramäki, Mikko Kivelä, Jukka-Pekka Onnela, Kimmo Kaski, and János Kertész. Generalizations of the clustering coefficient to weighted complex networks. *Phys. Rev. E*, 2007.

[125] Odnan Ref Sanchez, Simone Ferlin, Cristel Pelsser, and Randy Bush. Comparing machine learning algorithms for bgp anomaly detection using graph features. In *CoNEXT*, 2019.

[126] Kevin Hoarau, Pierre Ugo Tournoux, and Tahiry Razafindralambo. Suitability of graph representation for bgp anomaly detection. In *IEEE LCN*, 2021.

[127] Nabil Al-Rousan, Soroush Haeri, and Ljiljana Trajković. Feature selection for classification of bgp anomalies using bayesian models. In *2012 International Conference on Machine Learning and Cybernetics*. IEEE, 2012.

[128] Iñigo Ortiz de Urbina Cazenave, Erkan Köşlük, and Murat Can Ganiz. An anomaly detection framework for bgp. In *2011 Innovations in Intelligent Systems and Applications*. IEEE, 2011.

[129] Sharad Agarwal, Chen-Nee Chuah, Supratik Bhattacharyya, and Christophe Diot. The impact of bgp dynamics on intra-domain traffic. *ACM SIGMETRICS*, 2004.

[130] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, 1992.

[131] Chris Cheadle, Marquis P Vawter, William J Freed, and Kevin G Becker. Analysis of microarray data using z score transformation. *The Journal of molecular diagnostics*, 2003.

[132] Rafael A Irizarry, Bridget Hobbs, Francois Collin, Yasmin D Beazer-Barclay, Kristen J Antonellis, Uwe Scherf, and Terence P Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 2003.

[133] Jürgen Bajorath. Integration of virtual and high-throughput screening. *Nature Reviews Drug Discovery*, 2002.

[134] Paweł Szymański, Magdalena Markowicz, and Elżbieta Mikiciuk-Olasik. Adaptation of high-throughput screening in drug discovery—toxicological screening tests. *International journal of molecular sciences*, 2011.

[135] Jean-Luc Starck and Jerome Bobin. Astronomical data analysis and sparsity: From wavelets to compressed sensing. *Proceedings of the IEEE*, 2009.

[136] JW Brault and OR White. The analysis and restoration of astronomical data via the fast fourier transform. *Astronomy and Astrophysics*, 1971.

[137] Ziyi Chen, Liai Deng, Yuhua Luo, Dilong Li, José Marcato Junior, Wesley Nunes Gonçalves, Abdul Awal Md Nurunnabi, Jonathan Li, Cheng Wang, and Deren Li. Road extraction in remote sensing data: A survey. *International journal of applied earth observation and geoinformation*, 2022.

[138] Walter D Goldberger, Benjamin Grinstein, and Witold Skiba. Distinguishing the higgs boson from the dilaton at the large hadron collider. *Physical review letters*, 2008.

[139] Sciencealert. Less Than 1% of Large Hadron Collider Data Ever Gets Looked at, 2018. https://www.sciencealert.com/over-99-percent-of-large-hadron-collider-particle-collision-data-is-lost.

[140] Zheng Zhang, Ying Zhang, Y Charlie Hu, and Z Morley Mao. Practical defenses against bgp prefix hijacking. In *ACM CoNEXT*, 2007.

[141] Zhenhai Duan, Xin Yuan, and Jaideep Chandrashekar. Constructing inter-domain packet filters to control ip spoofing based on bgp updates. In *IEEE INFOCOM*, 2006.

[142] Matthew Caesar and Jennifer Rexford. Bgp routing policies in isp networks. *IEEE network*, 2005.

[143] BGPq4. Bgpq4, 2019. https://github.com/bgp/bgpq4.

[144] IRRpt. Irrpt. https://github.com/6connect/irrpt, 2006.

[145] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014.

[146] CAIDA. Mapping autonomous systems to organizations: Caida's inference methodology, 2020. https://www.caida.org/archive/as2org/.

[147] RIPE. RIPE RIS Live, 1. https://ris-live.ripe.net/.

[148] Ricardo Oliveira, Beichuan Zhang, Dan Pei, Rafit Izhak-Ratzin, and Lixia Zhang. Quantifying path exploration in the internet. In *ACM IMC'06*, 2006.

[149] Nick Feamster, Jay Borkenhagen, and Jennifer Rexford. Guidelines for interdomain traffic engineering. *ACM SIGCOMM CCR*, 2003.

[150] Steve Uhlig and Olivier Bonaventure. Designing bgp-based outbound traffic engineering techniques for stub ases. *ACM SIGCOMM CCR*, 2004.

[151] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu, and Lixia Zhang. An Analysis of BGP Multiple Origin AS (MOAS) Conflicts. In *ACM IMW*, 2001.

[152] Kwan-Wu Chin. On the characteristics of bgp multiple origin as conflicts. In *IEEE ATNAC*, 2007.

[153] CitizenLab. A case study of the China Telecom incident, 2012. https://citizenlab.ca/2012/12/.

[154] Ars Technica. Russian-controlled telecom hijacks financial services' internet traffic, 2017. https://arstechnica.com/security/2017/04/.

[155] CAIDA. Routeviews prefix to as mappings dataset (pfx2as) for ipv4 and ipv6, 2019. https://www.caida.org/catalog/datasets/routeviews-prefix2as/.

[156] Geoff Huston and George Michaelson. Validation of route origination using the resource certificate public key infrastructure and route origin authorizations, 2012.

[157] NTT. Rpki records, 2024. https://rpki.gin.ntt.net/api/export.json.

[158] IRR. Internet routing registry, 2024. https://irr.net/.

[159] Z Morley Mao, Lili Qiu, Jia Wang, and Yin Zhang. On as-level path inference. In *ACM SIGMETRICS*, 2005.

[160] Matthew Edman and Paul Syverson. As-awareness in tor path selection. In *ACM CCS*, 2009.

[161] CAIDA. Bgp2go, 2024. http://nids.caida.org:45000/cgi-bin/peerstats.sh.

[162] Pei-chun Cheng, Xin Zhao, Beichuan Zhang, and Lixia Zhang. Longitudinal study of bgp monitor session failures. *ACM SIGCOMM CCR*, 2010.

[163] Lan Wang, Malleswari Saranu, Joel M Gottlieb, and Dan Pei. Understanding bgp session failures in a large isp. In *IEEE INFOCOM*, 2007.

[164] Jong Han Park, Dan Jen, Mohit Lad, Shane Amante, Danny McPherson, and Lixia Zhang. Investigating occurrence of duplicate updates in bgp announcements. In *IEEE PAM*, 2010.

[165] CAIDA. As rank, 2023. https://asrank.caida.org/.

[166] Anant Shah, Romain Fontugne, and Christos Papadopoulos. Towards characterizing international routing detours. In *Proceedings of the 12th Asian Internet Engineering Conference*, pages 17–24, 2016.

[167] MANRS. Mutually Agreed Norms for Routing Security (MANRS), 2024. https://www.manrs.org/about/.

[168] MANRS Rostelecom. Not just another BGP Hijack, 2020. https://www.manrs.org/2020/04/not-just-another-bgp-hijack/.

[169] Cloudflare. How we detect route leaks and our new Cloudflare Radar route leak service, 2022. https://blog.cloudflare.com/route-leak-detection-with-cloudflare-radar/.

[170] Ben Du, Gautam Akiwate, Thomas Krenc, Cecilia Testart, Alexander Marder, Bradley Huffaker, Alex C. Snoeren, and KC Claffy. IRR Hygiene in the RPKI Era. In *PAM*, 2022.

[171] Ben Du, Katherine Izhikevich, Sumanth Rao, Guatam Akiwate, Cecilia Testart, Alex C. Snoeren, and kc claffy. Irregularities in the internet routing registry. In *ACM IMC*, 2023.

[172] Massimiliona Stucchi. Do we still need the IRR? An analysis and comparison of IRR data across databases, 2022. https://ripe85.ripe.net/wp-content/uploads/presentations/71-10-RIPE85-IRRAnalysis.pdf.

[173] Yet another BGP hijacking towards AS16509, 2022. https://mailman.nanog.org/pipermail/nanog/2022-August/220320.html.

[174] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, 2018.

[175] BGPview. BGPview, 2024. https://bgpview.io/".

[176] Ignacio Castro, Juan Camilo Cardona, Sergey Gorinsky, and Pierre Francois. Remote peering: More peering without internet flattening. In *ACM CoNEXT*, 2014.

[177] Fabricio Mazzola, Pedro Marcos, Ignacio Castro, Matthew Luckie, and Marinho Barcellos. On the latency impact of remote peering. In *IEEE PAM*, 2022.

[178] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks, 2016.

[179] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G. Andersen. SCION: Scalability, Control, and Isolation on Next-Generation Networks. In *S&P*, 2011.

[180] MicroBGP. Micro bgp suite: The swiss army knife of routing analysis, 2021. https://git.doublefourteen.io/bgp/ubgpsuite.

[181] BGPscanner. New mrt-bgp reader six times faster than its predecessors, 2018. https://blog.apnic.net/2018/11/29/new-mrt-bgp-reader-six-times-faster-than-its-predecessors/.

[182] MRTparse. Mrtparse, 2022. https://github.com/t2mune/mrtparse.

[183] PostgreSQL Global Development Group. Postgresql, 2024. https://www.postgresql.org.

[184] Yehuda Afek, Omer Ben-Shalom, and Anat Bremler-Barr. On the structure and application of bgp policy atoms. In *ACM IMC*, 2002.

[185] OpenAI. Chatgpt (4o), 2024. https://chat.openai.com.

[186] Jing Yu Koh, Daniel Fried, and Ruslan Salakhutdinov. Generating images with multimodal language models, 2023. URL https://arxiv.org/abs/2305.17216.

[187] GitHUB. Copilot, 2024. https://github.com/features/copilot.

[188] Yajie Zhou, Nengneng Yu, and Zaoxing Liu. Towards interactive research agents for internet incident investigation. In *ACM HotNets*, 2023.

[189] Prakhar Sharma and Vinod Yegneswaran. Prosper: Extracting protocol specifications using large language models. In *ACM HotNets*, 2023.

[190] Qiao Xiang, Yuling Lin, Mingjun Fang, Bang Huang, Siyong Huang, Ridi Wen, Franck Le, Linghe Kong, and Jiwu Shu. Toward reproducing network research results using large language models. In *ACM HotNets*, 2023.

[191] Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, Ye Tian, and Sujian Li. Restgpt: Connecting large language models with real-world restful apis, 2023. URL https://arxiv.org/abs/2306.06624.

[192] Next-Generation Database Interfaces: A Survey of LLM-based Text-to SQL. Zijin, hong1 and zheng, yuan2 and qinggang, zhang2 and hao, chen2 and junnan, dong2 and feiran, huang1 and xiao, huang, 2024. https://arxiv.org/html/2406.08426v1.

[193] David Clark and Cecilia Testart and Matthew Luckie and kc claffy. A path forward: Improving Internet routing security by enabling zones of trust. *Journal of Cybersecurity*, 2024.

[194] U.S. Federal Communications Commission. NOTICE OF PROPOSED RULEMAKING, In the Matter of Reporting on Border Gateway Protocol Risk Mitigation Progress and Secure Internet Routing, 2024. https://docs.fcc.gov/public/attachments/DOC-402609A1.pdf.

Améliorer la visibilité du routage dans l'Internet publique

Candidat: Thomas Alfroy
Encadrants: Pr. Cristel Pelsser
Dr. Thomas Holterbach
Dr. Pascal Mérindol

Préambule

Ce document résume les contributions effectuées par le candidat lors de sa thèse, encadrée par l'école doctorale MSII.

I) Introduction

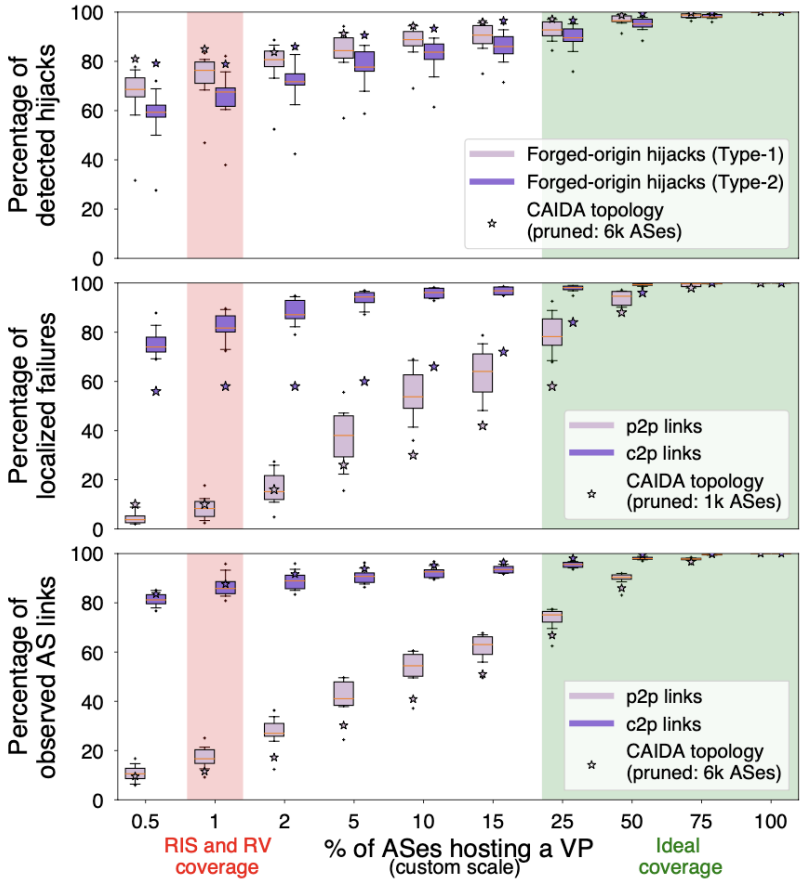
Internet est un système complexe opéré par différentes entités administratives appelées *Système Autonomes* (ou AS en anglais). Les différents AS participent au routage, c'est-à-dire au processus consistant à trouver le meilleur chemin d'un point à l'autre de l'Internet. Le protocole chargé d'assurer le routage dans l'Internet est *BGP* (Border Gateway Protocol). Bien que ce système fonctionne sans problèmes majeurs la majorité du temps, il arrive que certains événements inopinés perturbent son bon fonctionnement. Par exemple, une panne physique sur un lien connectant deux AS perturbe la connectivité, puisque tous les chemins empruntant ce lien sont désormais indisponibles et BGP doit en trouver un autre. Un attaquant peut également décider de manipuler BGP de sorte à détourner le trafic de sa destination légitime. Il est indispensable pour les opérateurs réseau de parvenir à détecter et résoudre ces anomalies sous peine de perdre leur connectivité vers un portion de l'Internet, résultant d'une insatisfaction de la part de leurs clients et d'une dégradation de leur réputation. Pour ce faire, les opérateurs utilisent des outils de contrôle du routage dans l'Internet. Ces outils se basent sur les données de routage BGP, collectés et mis à disposition par des plateformes de collecte publique, comme RIPE Routing Information Service (RIS) ou RouteViews (RV). Cependant, ces plateformes font face aujourd'hui à différents problèmes qui limitent leur efficacité d'utilisation par les opérateurs et les chercheurs.

II) Description du problème

Les plateformes de collecte de données BGP collectent des données depuis des *Vantage Points* (VP). Un VP est un routeur au sein d'un AS qui accepte d'envoyer ses meilleures routes vers un *collecteur de route*. Les données collectées depuis l'ensemble des VPs sont ensuite concentrées dans une base de données, que les utilisateurs peuvent utiliser afin d'effectuer diverses analyses de mesure. Cependant les plateformes de collecte font face aujourd'hui à deux limitations, dont les conséquences sont conflictuelles.

Les données collectées sont incomplètes. Les données BGP collectées par les plateformes sont vastement utilisées pour diverses analyses de mesure, ou pour effectuer de la détection d'attaques. Malheureusement, parce que BGP est un protocole de routage qui n'envoie que sa meilleure route vers chaque possible destination, chaque VP n'a pas une vue globale de ce qu'il se passe dans l'Internet. Les plateformes de collecte de données BGP récupèrent les données depuis moins de 2% des AS opérant dans l'Internet. Ainsi, ces plateformes n'ont pas un couverture complète, elles manquent donc inévitablement certains évènements.

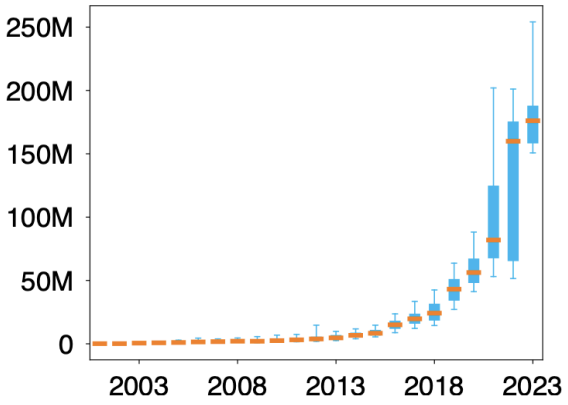
La première contribution de cette thèse consiste à estimer les trous de visibilité de ces plateformes. Cependant, puisqu'il est impossible de précisément savoir ce que l'on manque depuis des VPs qui ne sont pas déployés, nous avons choisi d'estimer cet écart entre les dynamiques réelles de routage dans l'Internet et ce que l'on peut observer depuis les données collectées à l'aide de simulations réalistes. Nous avons simulé plusieurs écosystèmes de routage de l'Internet dans lesquels nous avons généré divers évènements. Nous avons ensuite étudié si ces évènements étaient correctement observables par un ensemble de VP dont la couverture correspond à la couverture actuelle des plateformes de collecte.



La figure ci-dessus présente les résultats de ces expériences. La zone en rouge correspond à la couverture des plateformes de collectes actuelles (<2%). Selon, nos simulation, nous ne détectons actuellement que 16% des liens inter-AS dits "peer-to-peer" qui existent dans l'Internet, seulement 12% des pannes sur les liens

“peer-to-peer” peuvent être précisément localisées, et 30% des attaques de routage de type “BGP hijack” restent indétectables. Nous pouvons donc en conclure qu’il est primordial d’augmenter drastiquement la couverture de ces plateformes de collecte.

Le volume de données collectées empêche le déploiement de nouveaux VPs. Puisque l’Internet est un système en constante évolution, les données de routage collectées augmentent drastiquement au cours du temps. Le nombre de VPs à augmenté au fil des années, mais l’Internet s’élargissant également, la proportion d’AS déployant un VP est restée la même. De la même manière, la quantité de données collectées par chaque VP a également augmenté. Ces deux phénomènes combinés résultent en une augmentation quadratique du volume de données collecté par les plateformes de collecte. comme on peut le voir sur la figure ci-dessous.



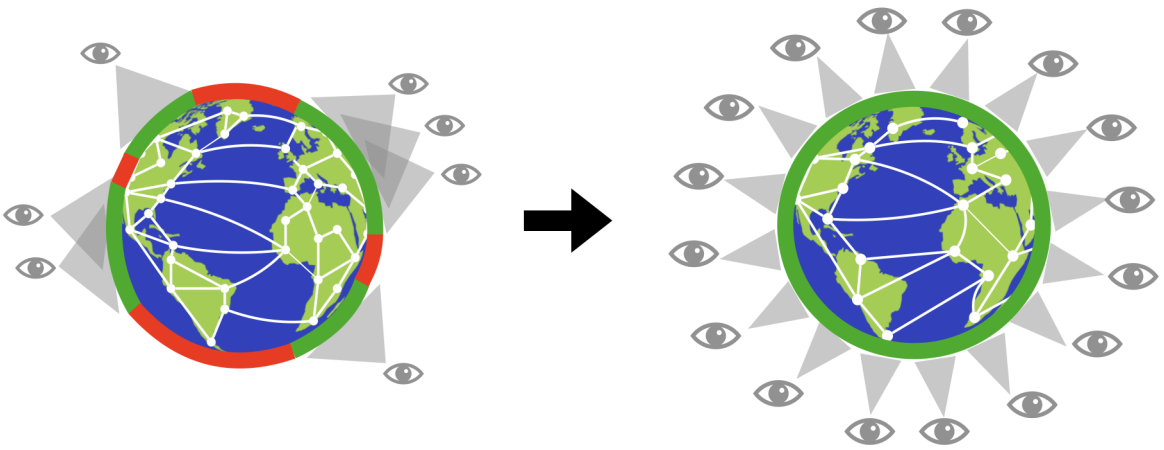
Le volume de données collecté à tant augmenté au cours des vingt dernières années, qu’il correspond maintenant à un TéraOctet de données brutes collectées chaque jour. Ce volume toujours croissant empêche une augmentation drastique de la couverture des VPs, et donc de découvrir de nouvelles informations à propos du routage dans l’Internet.

Résoudre le problème de visibilité dans l’Internet est un problème complexe. Augmenter la visibilité des routes BGP dans l’Internet nécessite d’augmenter le nombre de VPs depuis lesquels les données sont collectées. Cependant, même avec la faible couverture d’aujourd’hui, le volume de données est ingérable, demandant aux opérateurs des plateformes de collecte de refuser les nouvelles demandes d’ajout de VPs, empêchant ainsi l’expansion de ces plateformes. Il est donc nécessaire, afin d’améliorer la sécurité dans l'Internet, de trouver de nouvelles stratégies pour pallier au manque de visibilité.

III) Description de la solution

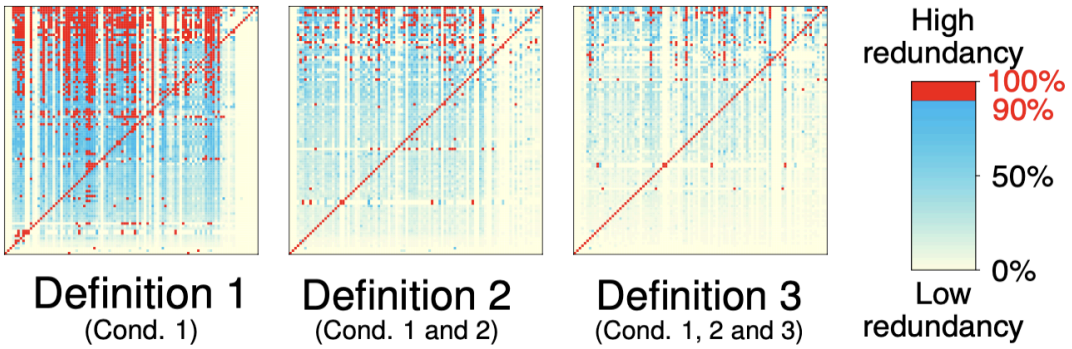
Au cours de cette thèse, le doctorant a développé GILL, un système de collecte permettant de supporter jusqu’à 20 fois plus de VPs que les systèmes de collecte actuels. La solution proposée utilise une observation clé: les données BGP sont très redondantes. GILL utilise cette propriété pour identifier quelles proportions des données sont redondantes et peuvent être supprimées sans perte d’information. Le système utilise donc une approche dite “overshoot-and-discard”. Ce nouveau paradigme consiste

à collecter des données depuis un maximum de VPs, puis de supprimer la portion des données qui aura été considérée comme la moins utile.



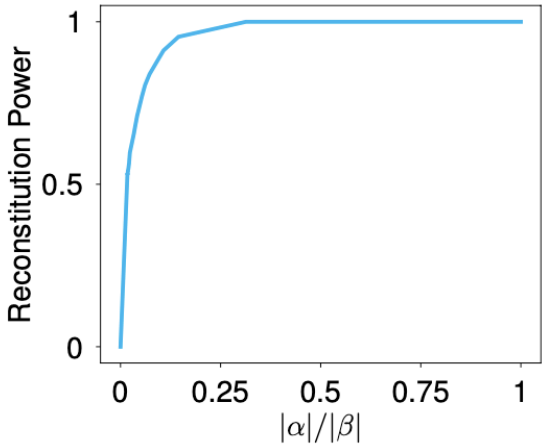
La stratégie employée permet donc le déploiement de nouveaux VPs, sans pour autant augmenter le volume de données à gérer. Comme dit plus haut, cette approche se base sur une propriété importante des données BGP: sa grande redondance.

Les données BGP sont très redondantes. Il est difficile de définir précisément ce qu'est la redondance dans les données BGP, puisque chaque utilisateur peut avoir un objectif très différent de celui de son homologue. Lors de cette thèse, nous avons proposé une série de trois définitions simples de plus en plus strictes permettant de montrer l'importante redondance entre les BGP updates, c'est-à-dire les messages BGP permettant aux AS d'échanger leurs routes. Ces définitions, bien que simplistes permettent d'illustrer le haut taux de redondance dans les données BGP. Les résultats sont présentés sur la figure ci-dessous, ou la couleur d'une case correspond au taux de redondance entre le VP dans l'axe des X et le VP dans l'axe des Y.



Avec la première définition, 97% des BGP updates collectés sont redondants avec au moins un autre. Ce nombre décroit à 70% avec la seconde définition et à 67% avec la dernière. Selon la première définition, 71% des VP ont au moins 90% de leurs updates qui sont redondants avec ceux d'un autre VP. Ce nombre décroît avec une définition plus stricte, 37% avec la seconde et 28% avec la dernière. Ces chiffres illustrent l'importante quantité de redondance dans les données BGP et la possibilité pour notre algorithme de les exploiter.

GILL élimine les BGP updates redondants. GILL est un système implémentant une stratégie de collecte différente de celle employée aujourd'hui. Les plateformes actuelles collectent toutes les données depuis peu de VPs. GILL quant à lui collecte un sous-ensemble des données depuis davantage de VPs. GILL est donc développé pour éliminer la redondance dans les données BGP. Malheureusement, nous avons montré lors de cette thèse qu'éliminer la redondance en se basant sur une définition simpliste donne de mauvaises performances pour de nombreuses applications pratiques. Nous nous basons donc sur une nouvelle métrique que nous avons développée et appelée la *capacité de reconstitution,* dont voici l'intuition: s'il est possible de reconstituer un ensemble de BGP updates depuis un de ses sous ensemble, cela signifie que le sous ensemble contient les données les plus utiles et les données retirées sont les plus redondantes.
Gill se base sur une autre propriété. Les redondances observées à un instant données ont des chances très grandes d'être également observées dans le futur. Ainsi, GILL calcule l'ensemble de données non redondantes à un instant donné, construit des filtres permettant de les discriminer des données redondantes et déploie ces filtres sur les VPs. La figure ci-dessous présente les résultats de la capacité de reconstitution de GILL, l'axe des X correspond à la proportion de routes qui n'ont pas été supprimée et l'axe des Y correspond à la proportion des routes originales qui ont pu être reconstituées sur base de celles qui ont été conservées.



Dans la pratique, GILL est en mesure de filtrer 94% des données, tout en limitant la perte d'information. Ces résultats démontrent l'efficacité de cette définition de la redondance dans les données BGP. Nous avons également réalisé une implémentation de GILL, dont les performances sont affichées dans le tableau ci-dessous.

Number of peers		100	1000	10000
With filters (i.e., GILL)				
Update load (per hour)	Average (28K upd/h)	0%	0%	0%
	99th percentile (241K upd/h)	0%	0%	high
Without filters				
Update load (per hour)	Average (28K upd/h)	0%	0%	39%
	99th percentile (241K upd/h)	0%	32%	high

Ce tableau montre la proportion de messages BGP perdus par notre implémentation, en fonction de la charge reçue. Nous pouvons clairement voir que quelle que soit la charge de données reçue, GILL (avec filtres déployés) est en mesure de gérer au minimum 100 sessions BGP sur un simple serveur, et jusqu'à 10000 lorsque la charge est moyenne avec les filtres déployés.

IV) Evaluation

Lorsque GILL supprime des données, même redondantes, il est impossible de n'essuyer aucune perte d'information. Nous avons donc évalué la perte d'information liée à l'échantillonnage de GILL en utilisant divers objectifs pratiques, extraits de la littérature. Nous avons comparé l'approche actuelle des plateformes de collecte et notre approche en utilisant des simulations. Nous comparons l'information obtenue en collectant toutes les données depuis 2% des réseaux qui opèrent dans l'Internet (couverture actuelle) et l'information obtenue en collectant un sous ensemble de données (5%) depuis 50% des AS, en nous assurant que le volume de données collecté par les deux stratégies était le même. Nous présentons les résultats pour trois analyses de mesures:
Détection des liens inter-AS: cet objectif consiste à observer autant de liens inter-AS que possible. Cela permet d'améliorer notre connaissance de la topologie de l'Internet ainsi que de son routage en général. Avec l'approche actuelle, nous sommes en mesure de détecter 20% des liens inter-AS, contre 63% avec GILL, soit plus de trois fois plus.
Localisation des pannes inter-AS: cet objectif consiste à localiser précisément les pannes inter-AS. Ces pannes peuvent être liées à des pannes physiques, ou des erreurs de configuration. Avec l'approche actuelle, 37% des pannes peuvent être localisées précisément, contre 80% avec GILL.
Détection des attaques de routage: cet objectif consiste à détecter autant d'attaques de routage que possible. Dans notre cas, nous nous focalisons sur un type spécifique d'attaque: les *Forged-Origin Hijacks*. Avec l'approche actuelle, 72% de ces attaques sont détectées, contre 83% avec GILL
Les résultats de ces analyses sont montrés dans le tableau ci-dessous.

Coverage		**2%**			**10%**			**25%**			**50%**		
Data Collection Scheme		GILL	Rnd. VP	Best Case	GILL	Rnd. VP	Best Case	GILL	Rnd. VP	Best Case	GILL	Rnd. VP	Best Case
Updates retained / Anchor VPs		*18.0% / 17.0%*		100%/100%	*7.9% / 3.3%*		100%/100%	*5.4% / 1.3%*		100%/100%	*4.7% / 0.9%*		100%/100%
Use cases	Topology Mapping	14%	4%	20%	33%	7%	50%	42%	7%	69%	61%	16%	85%
	Failure Localisation	29%	11%	37%	61%	14%	81%	60%	25%	80%	80%	18%	92%
	Hijack detection	58%	53%	73%	73%	54%	87%	77%	59%	92%	82%	74%	96%

Nous pouvons en conclure que la stratégie employée par GILL permet d'améliorer les résultats de nombreuses analyses qui peuvent être faites sur les données BGP.

Nous avons également comparé GILL aux autres stratégies actuelles de collecte des données BGP. Nous assurons que notre comparaison soit juste en utilisant un volume de données qui soit exactement le même pour toutes les stratégies et mesurons la proportion d'événements détectables avec les données collectées par chacune de ces stratégies.

Random Updates: Cette stratégie consiste à utiliser une stratégie "overshoot-and-discard" aux données collectées. Cependant, contrairement à GILL ou les données sont filtrées sur base leur redondance, les filtres sont générés ici de manière aléatoire.
Random VPs: Cette stratégie consiste à collecter les données depuis un sous-ensemble des VPs disponibles, sélectionné de manière aléatoire.
AS-Distance: Cette stratégie consiste à ne collecter des données que depuis un sous-ensemble des VPs disponibles, en les sélectionnant sur base de leur distance. Les VPs qui sont loins dans la topologie de l'Internet ont plus de chances d'être sélectionnés ensemble.
Unbiased: Cette stratégie consiste à ne collecter des données que depuis un sous-ensemble des VPs disponibles, en les sélectionnant sur base de leurs caractéristiques (taille du réseau, place dans la topologie,...)

Ces différentes stratégies de collecte des données sont comparées sur cinq différents cas d'usage classiques des données BGP, que nous avons extrait de la littérature (papiers de recherche, blog post sur mailing list d'opérateurs réseaux,...).
Détection de chemins transitoires: Les chemins transitoires sont des routes BGP visibles pendant moins de cinq minutes, ce qui correspond au délai typique de convergence BGP, et peuvent être attribués, par exemple, à l'exploration de chemins. Nous nous concentrons sur tous les événements de chemins transitoires détectés au cours des 30 heures, soit un total de 859 000 événements.
Détection des attaques de type MOAS: Les préfixes MOAS sont annoncés par plusieurs AS distincts, en raison d'actions légitimes ou malveillantes. Nous utilisons la méthodologie de pour éliminer les faux positifs. Nous nous concentrons sur les 1587 événements MOAS observés au cours des 30 heures.
Création de la carte de l'Internet: Cela est utile, par exemple, pour déduire les politiques BGP ou les chemins AS. Pour chaque VP, nous traitons le premier dump RIB de septembre 2023 ainsi que les mises à jour collectées pendant les 30 périodes d'une heure. Nous nous concentrons sur les 687 000 liens AS distincts observés.
Détection des communautés d'action: Les communautés d'action sont associées aux actions d'ingénierie du trafic et sont les plus difficiles à observer. Nous considérons les 8683 communautés d'action fournies dans et observées au cours des 30 heures.
Détection des chemins inchangés: Les mises à jour BGP de chemins inchangés sont des annonces qui signalent uniquement un changement dans les valeurs de communautés, mais pas dans le chemin d'AS. Nous considérons les 263 000 mises à jour de chemins inchangés observées au cours des 30 heures.
Les résultats sont présentés dans le tableau ci-dessous:

	Sampling Scheme	*GILL*	*GILL*-simplified		Naive			
			GILL-upd	*GILL*-vp	Rnd. Upd.	Rnd. VP	AS-Dist.	Unbiased
Use cases	**Trans. path detection (*I*)**	96%	65%	75%	83%	75%	95%	89%
	MOAS detection (*II*)	95%	95%	45%	33%	35%	59%	46%
	Topo mapping (*III*)	90%	93%	61%	72%	41%	67%	38%
	Action Coms. detection (*IV*)	91%	95%	49%	79%	48%	41%	42%
	Unchanged-path Upd. detection (*V*)	87%	60%	80%	76%	74%	43%	61%

Nous calculons pour GILL et chaque méthode de référence la proportion d'événements qu'ils détectent ou de liens qu'ils observent, et rapportons les résultats dans le Tableau 2. Par exemple, dire que GILL détecte 95% des événements MOAS signifie que les échantillons de données de GILL permettent de détecter 95% des 1587 événements MOAS utilisés dans le benchmark. La cellule d'une méthode de référence est colorée en vert lorsque GILL surpasse cette méthode, en rouge si la méthode est meilleure, et en jaune si les deux obtiennent des performances similaires (±5%). Nous veillons à ce que les méthodes de référence traitent le même nombre de mises à jour que GILL, soit 6,7 % des mises à jour de RIS et RV. GILL surpasse chaque méthode de référence naïve pour chaque cas d'utilisation, et parfois de manière significative, par exemple, GILL détecte +62 %, +60 %, +36 % et +49 % de détournements MOAS (cas d'utilisation II) par rapport à Rnd.-Upd., Rnd.-VPs, Dist.-based et Unbiased, respectivement.

V) Conclusion

Dans cette thèse, nous avons démontré le manque de visibilité des plateformes actuelles de collecte de données BGP et illustré comment ce déficit de visibilité peut avoir un impact négatif sur les études de mesure courantes. Une solution naïve serait d'augmenter la couverture des VP ; cependant, cela est impossible, car les plateformes de collecte de données BGP et les utilisateurs font déjà face à des problèmes de gestion des données, même avec la couverture (faible) actuelle des VP. Nous avons identifié l'opportunité de tirer parti du haut niveau de redondance dans les données BGP pour réduire le volume de données à traiter tout en minimisant la perte d'information.

Publications:

The Next Generation of BGP Data Collection Platforms.
Thomas Alfroy, Thomas Holterbach, Thomas Krenc, KC Claffy and Cristel Pelsser,
In *Proc. of ACM SIGCOMM 2024*, Sydney, Australia.
🏆 SIGCOMM best paper award

A System to Detect Forged-Origin Hijacks.
Thomas Holterbach, Thomas Alfroy, Amreesh Phokeer, Alberto Dainotti and Cristel Pelsser, *In Proc. of USENIX NSDI 2024*, Santa Clara, USA.

Internet Science Moonshot: Expanding BGP Data Horizons.
Thomas Alfroy, Thomas Holterbach, Thomas Krenc, KC Claffy and Cristel Pelsser,
In *Proc. of ACM HotNets 2023*, Boston, USA.