Filtering the Noise to Reveal Inter-Domain Lies

Julián M. Del Fiore‡, Pascal Merindol‡, Valerio Persico*, Cristel Pelsser‡, Antonio Pescapé*
‡ICube, University of Strasbourg – France *University of Napoli Federico II – Italy

***Abstract*—On the Internet, routers of Autonomous Systems (ASes) have to determine their preferred inter-domain route, i.e. control path (CP), for each IP prefix. The traffic is then forwarded AS by AS, following a data path (DP) that should match the CP for the same prefix. The underlying implicit trust that ASes advertise the paths they use for packet forwarding may be misplaced. Network operators may tweak CPs and DPs to carry out inter-domain *lies* that are visible when the two paths differ. Lies can be either unintended, due to misconfigurations or technical limitations, or deliberate, e.g. for economical gain. While lies globally mitigate the ability to troubleshoot and understand the root cause of connectivity issues, detecting them is not a trivial task as the ground data is noisy.**

In this paper, we propose a modular framework to measure and correctly quantify the discrepancies between CPs and DPs. We define several rules to overcome specific sources of noise inducing mismatches (MMs), e.g., incomplete traces, sibling ASes, IXPs or third-party addresses in general. We leverage the PEERING testbed to conduct a measurement campaign at a scale never achieved before, and conclude that, while the upper bound of lies is significant, the lower bound is not negligible. This suggests that the noise interfering with collected traces is not the sole culprit for the MMs between CPs and DPs.

I. INTRODUCTION

The Internet is an interconnection of independent networks called Autonomous Systems (ASes) that rely on BGP, the border gateway protocol, for inter-domain routing. ASes determine, through the exchange of BGP announcements, preferred routes or control paths (CP) for all IP prefixes. On the other hand, at the data plane the traffic is forwarded hop-by-hop, AS after AS, following a data path (DP), until it reaches the destination. Considering how BGP works, network operators expect both CP and DP for each prefix to match (as does previous research [1], [2], [3], [4]), but have no further means to easily verify it. The implicit trust that ASes advertise the paths they use for packet forwarding may be misplaced. Network operators may tweak CPs and DPs to carry out inter-domain *lies* that are visible when the two paths differ.

The purpose of these inter-domain lies may be to redirect and intercept traffic, or hinder its tracking with consequences on the ability to troubleshoot connectivity issues. Moreover, these lies may violate the contract between two adjacent ASes, with potential subsequent legal retaliation. Lies may lay in the control plane, where the AS-path is manipulated [5], or in the data plane [6], [7] diverting the DP from the CP. Furthermore, lies can be divided in either *deliberate* or *unintended*. Deliberate lies may obfuscate traffic interceptions or be driven by economical interests (e.g. attracting traffic by promising interesting routes but using cheaper alternatives). On the other hand, unintended mismatches can result from incongruent logical and physical topologies, in particular when BGP sessions are not set on simple point-to-point inter-domain links. Others may be rooted in technical limitations, such as limited memory on the routers hampering the storage of the full routing table, i.e. resulting in a partial Forwarding Information Base (FIB).

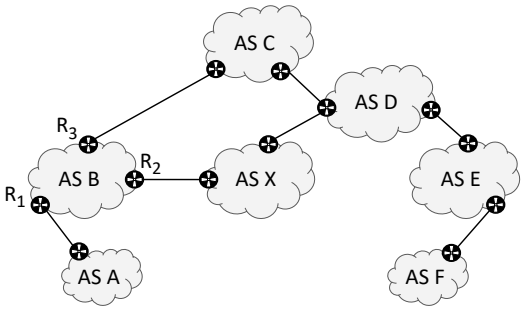


Fig. 1: Where is the lie? Being, A and F stub ASes; B, X and E transit providers; and C and D Tier-1 and Tier-2 ASes, respectively.

To illustrate some of these lies, let us rely on the topology in Fig. 1 and assume that Gao-Rexford policies are verified [8]. In all cases, we consider that data traffic is flowing from A towards a prefix owned by F. Although the traversed DP will *always* be equal to $ABXDEF$ the CP, obtained via BGP, will vary, as well as the reasons why it mismatches the DP.

Let us first consider that X learns the path $XDEF$ (towards the origin F). Since B and X are engaged in a peer-to-peer relationship, X does not export this path to B. Hence, B, and thus A, can only reach F via C: the CP for A is $ABCDEF$. However, B, that wishes to avoid paying for transit, assumes that X knows a path to reach F and forwards it the traffic, e.g. using a static route. If X does not filter any traffic it receives from B, then DP equals $ABXDEF$ and differs from CP. In this scenario, B carries out an *interested lie* against A and X.

Assuming now that B is a customer of X, then B learns two paths to reach F: via X or, as before, via C. Since both paths have the same length, then R_3 and R_2 opt for the paths via C and X, respectively. As R_1 has a shorter internal path to R_3 than to R_2, then A learns the same CP as before: the traffic should flow from R_1 to R_3 in B, and from there to C. However, because R_1 has a partial FIB and uses R_2 as default gateway, the traffic finishes exiting B via X, through R_2. In this second example, the exact same mismatch (MM) as before is now caused by a *technical limitation* at R_1.

Yet another cause of MMs is *AS poisoning*: an AS can interfere with a competitor AS by poisoning it, i.e by prepending the AS number (ASN) of the latter to the path. In such cases, the competitor AS finds itself already in the path

and rejects the BGP update, possibly incurring in a loss of revenue. For example, if E poisons C, then only X accepts the path advertised by D. Consequently, A finishes learning $ABXDECF$ as CP. Since C had been artificially added to the CP, the traffic does not actually traverse it, naturally leading to a mismatch between CP and DP. Other manipulations, such as AS deletions in CPs, can be used. As an example, if B advertises to A a path where X previously deleted D and E, then CP equals $ABXF$. However, in practice, DP crosses extra inter-domain links that do not appear in CP.

Our main objective is to detect lies and to understand their cause (technical limitations, AS poisoning, interested). However, a considerable practical challenge needs to be addressed first: the state-of-the-art **active measurement techniques are noisy**. Hence, since lies may be misinterpreted as noise, or vice versa, filtering this noise is imperative. In that sense, our main contribution in this paper is the proposal of a framework to reveal highly-potential lies by eliminating all sources of errors interfering with ground traces (discussed in Sec. II and Sec. III). However, as shown in the previous examples, the same discrepancy between CPs and DPs may actually be caused by different root causes and, thus, being able to pinpoint causes of lies is left as future work.

As a first step towards our ultimate goal, the contributions presented in this paper are:

1. We develop a framework computing multiple bounds of MM rates; each model tackling different sources of noise in the collection of CPs and DPs (Sec. IV).
2. We focus on the most complete model, that should remove all the noise. Though validation requires ground truth, a non-negligible amount of highly-potential lies is revealed.

Moreover, to the best of our knowledge, we are the first to:

- Deploy 8 vantage points (VPs) with co-located paths and full BGP feeds (i.e RIBs): six in the PEERING Testbed and two additional where we may also benefit from a privileged view on routing configurations (Sec. V).
- Carry out a long-term comparison between DPs and CPs (Sec. VI). We find that, usually, MMs do not vary in time.

Our main findings and final remarks are discussed in Sec. VII.

II. Background

The comparison of CPs and DPs has received little attention so far. In [3], Mao et al. find that Internet Exchange Points (IXPs), AS siblings, and ASes announcing IP prefixes for which they are not the real Origin AS (OAS) are predominant causes for MMs. In a follow-up work [4], they develop a systematic approach to correct inaccurate IP-to-AS mappings by reassigning the OAS of prefixes. Hyun et al. [1] also analyze the discrepancies between CPs and DPs and conclude that insertions of IXPs in the DPs and of ASes under the same ownership are the main cause that leads to MMs. However, in their study, incomplete traces are discarded and the comparison does not rely on the latest BGP updates, i.e. CPs and DPs are not synchronized. Zhang et al. [2] extract the mismatching fragments of CPs and DPs and show that the main pitfall of using traceroute in AS-level topology measurements originates from the appearance of IP addresses assigned from AS neighbors. However, their measurement platform suffers from the inability to ensure that the data and control plane VPs are co-located.

On the other hand, Hyun et al. [9] introduced the concept of Third-Party Addresses (TPAs), a source of noise affecting `traceroute`-like campaigns. A TPA is an IP address that is mapped to an AS that was actually not traversed by the traffic, and that results in false AS links. According to the authors, finding multiple TPAs in a row mapped to the same AS is unlikely, although possible. Their study concludes that TPAs are not common and that they do not distort AS maps significantly. A later analysis of Marchetta et al. [10] using IP timestamp options states the contrary. They find that consecutive TPAs are common, and may even entirely hide an AS from an AS-level path. However, a subsequent study from Luckie et al. [11] reports that most observed IPs in `traceroute` traces are from in-bound interfaces, thus on-path. They argue that techniques using IP timestamps are not reliable to detect TPAs. Finally, Ahmed et. al [12] proposed an offline method that tags up to two IPs that appear in a row as possible TPAs if they introduce an AS that either violates valley-free paths or translate into new AS relationships. Further work concerning detection of TPAs for correctly determining AS boundaries can be found in [13], [14], [15].

While these studies mainly blame the IP-to-AS mapping for the observed MMs, our work relies on *conservative* heuristics that remove the noise in the measurements and the mapping errors, e.g. all candidate TPAs are actually inferred to be TPAs. The MMs we find after having applied our filters show that the IP-to-AS mapping is not the only culprit for them.

III. Problem Statement

Let $\mathtt{CP}$ and $\mathtt{DP}$ be sequences of ASNs that denote respectively the control path and the data path for a given IP prefix:

$$\cdot \begin{cases} \mathtt{CP} = \mathtt{C}_1 \ \mathtt{C}_2 \ \ldots \ \mathtt{C}_i \ \ldots \ \mathtt{C}_n \\ \mathtt{DP} = \mathtt{D}_1 \ \mathtt{D}_2 \ \ldots \ \mathtt{D}_i \ \ldots \ \mathtt{D}_m \end{cases}$$

There exists a MM between $\mathtt{CP}$ and $\mathtt{DP}$, if $\exists \mathtt{j} \leq min(\mathtt{n}, \mathtt{m}) \mid \mathtt{C}_\mathtt{j} \neq \mathtt{D}_\mathtt{j}$. The rank $\mathtt{j}$ denotes the first position where the sequences differ. From a theoretical point of view, evaluating the consistence between CPs and DPs may appear a trivial task: the only requirement is to know for each analyzed prefix (e.g., a /24 IP address space) the path advertised through BGP routing announcements and the actual forwarding route in use towards the same prefix.

In practice, CPs and DPs are not so trivial to compare. To start, **synchronizing** both paths in space and time is mandatory. While achieving time synchronization is simple, and can be done just relying on timestamps extracted from the measurements (Sec. IV-B), space-synchronization actually depends on the measuring platform (Sec. V). Both CPs and DPs, respectively obtained from BGP peers and VPs, need to be measured in the same space, i.e. collected within the same

TABLE I: Summary of issues that alter CPs/DPs. *Sub*, *Ins*, and Del refer to substitutions, insertions, and deletion, respectively.

Issue	CP/DP	Effect	Path Distortion
AS siblings	CP/DP	*Sub*	A B C_0 **D_1** E
	CP/DP	*Ins*	A B **C_1** **C_2** D_0 E
TPAs	DP	*Ins*	A B **x** C_0 D_0 E
Wildcards	DP	*Del*	A B * * * * E

local network. In these cases, we refer to the VP as a co-located VP. In a co-located VP, in theory, DPs should match CPs for all destinations: a disagreement between both would mean that either noise affected the measurements or a lie took place. However, if the VP is not co-located, the VP does not share the same control plane as its peer, and the two paths may MM, but for valid reasons. For our analysis, we relied on 8 co-located VPs, six provided by the PEERING testbed [16] and two additional where we have privileged access. In addition to all this, DPs and CPs do not natively come at the same **semantic level**: the former are collected at the ground IP level, while the latter are provided by BGP at the AS level. Therefore, `traceroute` data has to be converted, with the use of an IP-to-AS mapping function, into AS-level paths.

Carrying out a comparison between CPs and DPs is not straightforward, even when the previous requirements are fulfilled: raw traces and mapping functions may suffer from both *inherited and inherent limitations*. Inherited limitations account for the ones generated by the IP-to-AS mapping process: IP addresses may either fail to be mapped i.e. the mapping is undefined, or be mapped to multiple ASes, e.g. due to organizations that use interchangeably the ASNs of the AS siblings they own [3]. On the other hand, inherent limitations are those that arise as a product of the measuring technique. For example, `traceroute` may provide both incomplete traces (i.e. include unresponsive hops [3], [17], [18]) and unreliable data (including TPAs [12], [10] possibly due to IXPs [1], [19] or AS boundary allocation policies [2]).

Accordingly, DPs may expose **AS substitutions**, **AS deletions** (generating wildcards, shown as '$*$'), and **AS insertions**, due to (i) practices involving the use of sibling ASNs, (ii) unresponsive hops in `traceroute` (or undefined mapping), and (iii) the presence of TPAs. Furthermore, CPs are also susceptible to suffer from substitutions and insertions due to AS siblings and prepending, but are not affected by other kind of noise (i.e. AS poisoning is considered a type of lie).

Table I summarizes the mentioned issues, reporting their effects and highlighting the distortion they generate, with respect to the *actual path* equal to A B C_0 D_0 E. The sibling issue may result in the substitution of an AS with one of its siblings (see line 1, where D_1 belongs to the same organization as D_0) or introduce new ASes (see line 2, with C_1 and C_2 belonging to the same organization of C_0). The presence of a TPA (always indicated with small letters) may lead to the insertion in the DP of an AS, `x` in this case (see line 3), which is actually not traversed. Finally, both unresponsive hops in `traceroute` traces (e.g., due to AS-scale configurations filtering expired probes) and undefined mapping generate wildcards, that may result in AS deletions (line 4).

Despite the presence of the above limitations, `traceroute` is still the most efficient and widespread tool for debugging the data plane and retrieving DPs. Consequently, MMs between CPs and DPs can only be correctly quantified if the shortcomings due to inherited and inherent limitations are jointly evaluated. In other words, as a first step towards the detection of lies, a framework that is able to take into account all sources of noise is needed.

IV. A Framework To Detect Inter-Domain Lies

This section describes the modular framework we propose to detect inter-domain lies, by filtering noise affecting the comparison of CPs and DPs. We first introduce at a high level the models we design in Sec. IV-A and then analyze their main blocks in Sec. IV-B, IV-C, and IV-D.

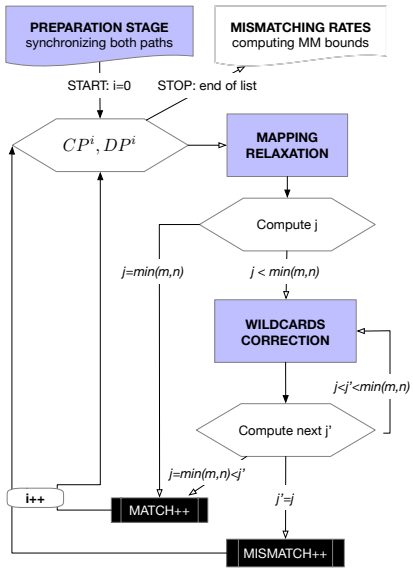


Fig. 2: Our modular framework: from data preparation towards mapping relaxations and wildcards corrections.

A. Our Modular Framework At a Glance

As illustrated in Fig. 2, the core of our framework consists in three stages (the violet boxes) that extensively manipulate CPs and DPs, and additional blocks that represent the logic and loops needed to carry both the noise filtering and the comparisons.

The **preparation stage** synchronizes CPs and DPs and translates the latter from the IP to the AS level. In the two remaining blocks, heuristics are implemented as *path-rewriting rules* that correct both DPs and CPs, mitigating the effects of the noise introduced by AS deletions, insertions, and substitutions. In particular, the **mapping relaxation** stage addresses limitations due to AS siblings and TPAs using two rules, namely `SIB` and `TPA`. The former relies on an AS-to-Organization mapping function, while the latter decides whether candidate TPAs are truly TPAs and replaces them with wildcards. Two variants exist for the latter, either `strictTPA` or `looseTPA`, which only differ in the conditions they request to confirm or not the candidate TPAs.

TABLE II: Summary of the correction rules. *Sub* is a substitution operation while the *Del* action performs a deletion.

Stage	Rule	Issue addressed	Action
Mapping Relaxation	SIB	Siblings and duplicates	*Sub/Del*
	looseTPA	Third-party addresses	*Sub*
	strictTPA	Third-party addresses	*Sub*
Wildcards Correction	match*	Wildcards	*Sub*
	nomatch*	Wildcards	*Del*

On the other hand, the **wildcards correction** stage compensates for AS deletions in DPs. Indeed, since wildcards result in incomplete paths, the incomplete sequences can either be substituted with their respective CP diverging sequence or just ignored (i.e. wildcards are deleted when no substitution is possible), as applied by rules `match*` and `nomatch*` respectively. This correction stage embeds the comparison at index j to iteratively use ASNs in the CP to complete the DP. Table II summarizes the rules applied in each of the blocks after the preparation stage, reporting the addressed issues as well as the actions taken to overcome them. Notably, the rules inside each block *are not commutative*. However, thanks to its modular design, our framework can be easily adapted to modify the rules in use, and to reorder them. While the preparation stage and the wildcards correction are mandatory, the mapping relaxation is optional: if the IP-to-AS mapping is assumed to be immune to noise, no complementary rules relaxing, and thus correcting it, are required.

In conclusion, depending on whether the mapping relaxation stage is used or not, which rules are applied and in which order they are implemented inside the blocks, **several models can be designed to explore the spectrum of path comparison**, as shown in Table III. We will use the name of each *model* to refer to either their design, or their mismatching rates (i.e. their resulting *bounds*). In particular, the *Lower* model is expected to report, in its *Lower* bound, less MMs than the *Restricted* one, not only because of the more conservative ordering of the rules applied in the mapping relaxation stage, but mostly because the latter implements a strict (exclusive), rather than a loose (permissive), `TPA` rule. While the *Lower* bound has complete freedom to introduce changes in DPs, the *Restricted* bound can only apply changes when a candidate TPA is not surrounded by neither wildcards or other candidate TPAs.

B. Preparation Stage

The preparation stage is fed with a set of raw CPs and DPs, and outputs a pre-processed AS-formatted list of (CP^i, DP^i) couples by: (i) synchronizing DPs and CPs, i.e. coupling each DP to a specific CP; (ii) IP-to-AS mapping each IP address appearing in the IP-level raw DPs; (iii) pre-processing each couple to purge them from minor mapping limitations.

1) Synchronizing DPs and CPs: each DP obtained running `traceroute` is associated to the CP of the longest matching prefix that covers the target IP in the last RIB dumped before the `traceroute` was run. This overall process results in a list of synchronized couples, where DP is still at the IP level.

2) IP-to-AS mapping: DPs are then converted into AS-level paths. We map each IP address in the DPs to the OAS of the longest matching prefix covering the IP[1]. However, `traceroute` traces may include private IPs, and usually unresponsive hops. Moreover, some IP addresses are not necessarily mapped to an unique and/or valid AS. OASes may include private, or more generally, prohibited ASNs (pASNs) that should not be advertised[2], and also AS sets. For all these cases, the mapping is undefined and we *conservatively* map them as wildcards ('$\star$'), that can be replaced by any value in the last post-processing stage.

TABLE III: Rules (columns) applied for different MM quantification models (rows). Roman numbers report the order of application of the rules. In contrast, ✗ denotes that the rule is not applied for the model.

	Mapping-Relaxation Stage			Wildcards-Correction Stage	
	SIB	looseTPA	strictTPA	match*	nomatch*
Raw	✗	✗	✗	✗	(i)
Upper	✗	✗	✗	(i)	(ii)
Restricted	(i)	✗	(ii)	(iii)	(iv)
Lower	(ii)	(i)	✗	(iii)	(iv)

3) Pre-processing couples: CPs may still include, and be affected by, **AS Sets**, **pASNs**, or **AS prepending**. Hence, a list of actions to purge them is required: (i) pASNs are removed when appearing at the end of a path; (ii) path couples with CPs containing AS sets or pASNs are discarded (less than 0.1% of the cases); (iii) repeated consecutive ASNs in CPs (AS prepending) are eliminated. Finally, (iv) in case of prematurely ending CPs (e.g. due to coarse grained prefixes) where DPs reveal extra path after the OAS, the remaining part of the path (if any) is trimmed.

After this stage, CPs and DPs are still subject to limitations introduced by AS siblings, TPAs or IXPs, and wildcard sequences. These sources of noise are filtered with the use of path-rewriting rules, as explained in the following sections.

C. Mapping Relaxation

Once the preparation stage has been completed, paths may still suffer from AS substitutions and insertions. Both are accounted for in this block, relying on two rules, namely `SIB` and `TPA`. The former acts in both CPs and DPs, linking AS siblings to unique representatives via an AS-to-Organization mapping function, while the latter modifies DPs by replacing inferred TPAs with wildcards. Although their respective mode of operation is based on distinct conditions, both rules relax the mapping, i.e. they re-map ASes by either grouping them by organizations, or turning them into wildcards. Note that this stage does not compare DPs and CPs, but rather simplifies both independently.

1) `SIB` rule: describing paths at an organization-level rather than at an AS-level solves AS substitutions and insertions introduced by AS siblings. We thus rely on an additional mapping function, called AS-to-ORG. Similarly to IP-to-AS mapping, the AS-to-ORG mapping consists in condensing paths: the former groups IPs into ASes whereas the latter

[1]The OASes of all prefixes were assumed to remain constant in the course of a day, and extracted from the first RIBs dumped every day.

[2]https://www.iana.org/assignments/as-numbers

gathers ASes into organizations. Note that the AS-to-ORG mapping has to be applied to both DPs and CPs to guarantee consistency.

Let $CH(\cdot)$ denote the AS-to-ORG mapping function and consider that each organization *Org* owns an AS sibling set $\mathbb{S}_{Org} = \{S_1, S_2, ..., S_N\}$ with $N \geq 1$. For each AS sibling set $\mathbb{S}_{Org}$, we arbitrarily define one of its siblings as the cluster head, denoted S. Then, $CH(S_i) = S,\ \forall S_i \in \mathbb{S}_{Org}$. Applying this additional mapping to the ASes in a path ensures that they are all mapped to the cluster head of the organization they belong to[3] while leaving wildcards unchanged (i.e., $CH(*) = *$). Additionally, possibly redundant ASNs resulting from multiple ASes being mapped to the same organization are purged. For example, let us consider a path P (either a DP or a CP) where each AS natively depicts the cluster head of its organization, except for $B_1, B_2 \in \mathbb{B}_{Org}$ such that $CH(B_i \neq *) = B$:

$$\texttt{P} = \texttt{A}\ \mathbf{B_1}\ \mathbf{B_2}\ * * \texttt{C D} \xrightarrow{\texttt{SIB}} \texttt{P} = \texttt{A}\ \mathbf{B}\ * * \texttt{C D}$$

SIB rule: CPs and DPs are AS-to-ORG mapped.

Input: $P \leftarrow CP\ or\ DP, CH(\cdot)$
for *all* $i \in [\![0, n[\![$ **do** // $n = len(P)$
  $P[i] \leftarrow CH(P[i])$
  if $P[i] = P[i-1]$ *and* $P[i] \neq *$ **then** // $n--$
    Delete $P[i]$
return P

Finally, note that the SIB rule keeps track of the number of IPs in each organization such that rules that are applied after it can take this parameter into account.

2) TPA rules: at an IP level, the traffic follows routes that are represented by the IPs of the incoming interfaces of the routers that are traversed towards the destinations. Although routers most likely respond to traceroute with the IP of their incoming interface, they can be configured differently: if the reply reports the IP address of another interface, specific inter-domain addressing allocation policies applied in IXPs or between incongruent remote BGP sessions may favor the occurrence of TPAs, that result in AS insertions in DPs.

To identify these AS insertions, the TPA rules check in DPs the amount of IP addresses that appear in a row and that are mapped to the same AS: if less than a threshold p, the AS is said to be *weakly represented* and, as such, becomes a candidate TPA. Furthermore, if certain conditions are verified, candidate TPAs are inferred to be real TPAs, and *conservatively* replaced with wildcards, rather than blindly and arbitrarily assigned to either the preceding or following AS.

Let $NH(\cdot)$ denote a function that takes an AS-level rank i as input, and returns the number of IP-level hops mapped to DP_i, i.e. the i^{th} AS-hop in the DP. If it holds that there exists a DP_i such that $NH(i) < p$, then we consider that this AS is included in the DP as a consequence of a candidate TPA. The $NH(\cdot)$ function is leveraged by our TPA rules, hereinafter assuming $p = 1$.

[3]We generated the AS-to-ORG mapping based on the Org_ID field of The CAIDA AS Organizations Dataset http://www.caida.org/data/as-organizations/20180703.as-org2info.txt. The cluster head for each organization was chosen as the first AS belonging to the sibling set that was found in available data.

TPA rules: the rules strictTPA and looseTPA differ only in that the triggering conditions, the latter being more permissive.

Input: $P \leftarrow DP, NH(\cdot), p$
for *all* $i \in [\![0, n[\![$ **do** // $n = len(P)$
  if $NH(i) < p$ **then**
    if *TPA rule* = *looseTPA* **then**
      $P[i] \leftarrow *$;
    else if *TPA rule* = *strictTPA* **then**
      if $NH(i \pm 1) \geq p$ *and* $P[i \pm 1] \neq *$ **then**
        $P[i] \leftarrow *$;
return P

While the looseTPA rule assumes *all* candidate TPAs are, in fact, real TPAs and replaces them with wildcards, the strictTPA rule only performs the replacement if: (i) both adjacent hops are *not* wildcards; (ii) both adjacent ASes are *not* candidate TPAs. To better understand the difference between both implementations, consider $\bar{\mathbb{P}}$ as a list of paths (DPs), where **x** and **y** are weakly represented ASes:

$$\bar{\mathbb{P}} = \begin{cases} \texttt{DP}_0 = \texttt{A B}\ \mathbf{x}\ \texttt{C D E} \\ \texttt{DP}_1 = \texttt{A B}\ \mathbf{x}\ \mathbf{y}\ \texttt{D E} \\ \texttt{DP}_2 = \texttt{A B}\ \mathbf{x}\ *\ \mathbf{y}\ \texttt{D E} \\ \texttt{DP}_3 = \texttt{A B}\ \mathbf{x}\ *\ \mathbf{x}\ \texttt{C D E} \end{cases}$$

The looseTPA and the strictTPA rules act as follows:

$$\bar{\mathbb{P}} \xrightarrow{\texttt{looseTPA}} \begin{cases} \texttt{DP}_0 = \texttt{A B}\ \mathbf{*}\ \texttt{C D E} \\ \texttt{DP}_1 = \texttt{A B}\ \mathbf{*}\ \mathbf{*}\ \texttt{D E} \\ \texttt{DP}_2 = \texttt{A B}\ \mathbf{*}\ *\ \mathbf{*}\ \texttt{D E} \\ \texttt{DP}_3 = \texttt{A B}\ \mathbf{*}\ *\ \mathbf{*}\ \texttt{C D E} \end{cases}$$

$$\bar{\mathbb{P}} \xrightarrow{\texttt{strictTPA}} \begin{cases} \texttt{DP}_0 = \texttt{A B}\ \mathbf{*}\ \texttt{C D E} \\ \texttt{DP}_1\texttt{, DP}_2\texttt{, and DP}_3 \text{ remain unchanged} \end{cases}$$

As shown in the examples above, the looseTPA rule is more conservative: with the least required evidence, *all* candidate TPAs are actually inferred to be, and turned into wildcards. Moreover, even when separated by unresponsive hops, the different appearances of the same AS are considered independent (see $\texttt{DP}_3$). On the other hand, strictTPA is less permissive and, since candidate TPAs and/or wildcards are considered logically exclusive, only $\texttt{DP}_0$ finishes being modified in the previous examples. Therefore, a model of our framework implementing looseTPA rather than strictTPA will have a lower MM rate, but may underestimate it.

D. Wildcards Correction

The last step required before being able to evaluate if DPs and CPs match is about solving AS deletions in DPs. Indeed, DPs may still be incomplete, i.e. include sequences of wildcards resulting from either unresponsive hops in traceroute, undefined IP-to-AS mapping or due to the previous application of the TPA rules. We will refer to the appearance of one or multiple wildcards in a row as a *wildcard sequence*. All wildcard sequences are bounded by two ASes: a *diverging AS* on the left and a *converging AS* on the right[4].

[4]Trailing wildcards that constitute an exception for the presence of a converging AS, are silently discarded as carrying no additional information.

The objective of the wildcard correction block is to *correct* DPs by replacing wildcard sequences with their respective CP sequence (rule `match*`). However, when substitutions cannot be performed, or simply systematically by the *Raw* model, wildcards are deleted (rule `nomatch*`). Both rules require knowing the AS-hop j where the first wildcard appears (right after the diverging AS) to be applied in each wildcard sequence.

To apply the `match*` rule, both the diverging and the converging ASes are required to appear in the CP. If so, and considering there is at least one intermediary AS in the CP sub-sequence between these two ASes, two possibilities may arise: (i) the number of ASes in the CP sub-sequence is smaller or equal to the length of the wildcard sequence in the DP or; (ii) the opposite. If (i) holds, the `match*` rule is able to correct the DP: the complete sequence of wildcards is substituted with the CP sub-sequence (extra wildcards, if any, being discarded); otherwise, the `match*` rule cannot rewrite the DP. In such cases, the DP may be further corrected with the `nomatch*` rule, that simply deletes the remaining wildcards and also the diverging AS when it matches the converging AS.

`match*`/`nomatch*` rules: while in rule `match*` wildcards are substituted (matched) with sequences of ASes in CPs, in rule `nomatch*` they are deleted.

`match*` rule $\Longrightarrow$ **Input:** $P \leftarrow DP, R \leftarrow CP, j$
  if $(\exists\ k > j \mid P[l] = * \ \forall\ l \in [\![j, k]\!]$ **then**
    if $(\exists\ i \in [\![j, k]\!] \mid P[k] = R[i])$ **then**
      Substitute $P[j], ..., P[k-1]$ with $R[j], ..., R[i-1]$
  return P

`nomatch*` rule $\Longrightarrow$ **Input:** $P \leftarrow DP, j$
  if $(\exists\ k > j \mid P[l] = * \ \forall\ l \in [\![j, k]\!]$ **then**
    Delete $P[j], ..., P[k-1]$
    if $P[j] = P[j-1]$ **then**
      Delete $P[j]$
  return P

In the following examples, let $\bar{\mathbb{P}}$ represent a list of DPs, each of which includes one or more sequences of wildcards, and that should be compared with the same control path `CP`:

$$\texttt{CP} = \texttt{A B C D E F G H}$$

$$\bar{\mathbb{P}} = \begin{cases} \texttt{DP}_1 = \texttt{A B C } \mathbf{\texttt{D * * * G}}\ \texttt{H} \\ \texttt{DP}_2 = \texttt{A B C } \mathbf{\texttt{D * G}}\ \texttt{H} \\ \texttt{DP}_3 = \texttt{A } \mathbf{\texttt{B * B}}\ \texttt{C D } \mathbf{\texttt{E * G}}\ \texttt{H} \end{cases}$$

If rules `match*` and `nomatch*` are consecutively applied (e.g., in an if/else condition), as for all models except *Raw*, then:

$$\bar{\mathbb{P}} \xrightarrow{\texttt{match*/nomatch*}} \begin{cases} \texttt{CP} = \texttt{DP}_1 = \texttt{A B C } \mathbf{\texttt{D E F G}}\ \texttt{H} \\ \texttt{CP} \neq \texttt{DP}_2 = \texttt{A B C } \mathbf{\texttt{D G}}\ \texttt{H} \\ \texttt{CP} = \texttt{DP}_3 = \texttt{A } \mathbf{\texttt{B}}\ \texttt{C D } \mathbf{\texttt{E F G}}\ \texttt{H} \end{cases}$$

Whereas in $\texttt{DP}_1$, rule `match*` replaces the three wildcards with the sub-sequence `E F` ($j = 5, k = 8, i = 7$), the substitution cannot be applied in $\texttt{DP}_2$ (the wildcards available are not enough). Moreover, even the subsequent use of rule `nomatch*` does not solve the difference between `CP` and $\texttt{DP}_2$. On the contrary, $\texttt{DP}_3$ matches `CP` after the first and second wildcard sequences are solved with rules `nomatch*` and `match*`, respectively.

The rules applied in this block are particularly *conservative* since each single wildcard can represent up to one entire AS (check $\texttt{DP}_3$ on the previous example). While this seems unlikely, it allows to minimize the MM rate since the number of ASes that can be inserted in DPs by replacing wildcard sequences, to overcome AS deletions, is maximized.

V. The Measurement Platform and our Campaign

In our measurement campaign, we leveraged the **PEERING testbed** [16] to deploy six VPs in addition to two homemade VPs in which we have also access to BGP dumps. This allowed us to reach a number of co-located VPs as never achieved before for this kind of analysis.

We collect the **CPs** from BGP speakers at the PEERING testbed and the homemade VPs (*hmX*). We focus on the peers reported in Table IV, which provide transit [20] and dump the full RIBs every 2 hours. On the other hand, we gather **DPs** with Scamper [21], running Paris-`traceroute` from VPs placed next to the gateway for the homemade VPs, or tunneling through the PEERING testbed up to the routers that provide the RIBs.

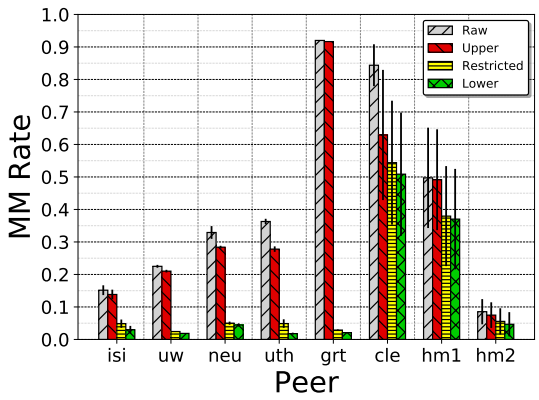
TABLE IV: Peers, that provided transit and full RIBs, used as VPs.

Peer	Organization	ASN	CP-DP match [%]
isi	Los Nettos	226	77.92
uw	University of Washington	101	77.93
neu	Northeastern University	156	76.84
uth	University of Utah	210	69.51
grt	GRNet	5408	77.93
cle	Clemson University	12148	77.93
hm1	University of Strasbourg	2259	77.94
hm2	RGnet, LLC	3130	77.90

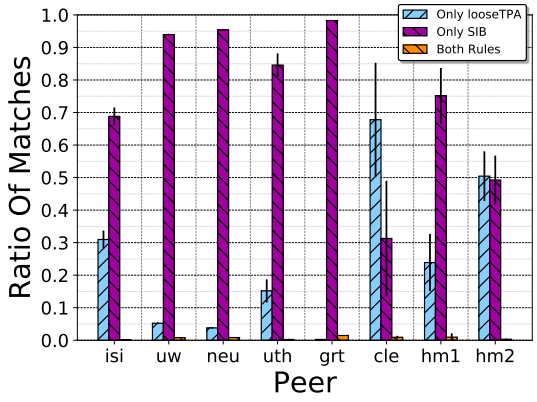
The measurement campaign was designed to run daily with 80k traces per day. For these traces, we chose the destinations by uniformly sampling /24 prefixes in blocks allocated by RIRs [22]. We pick one IP from each of these prefixes. However, for a fraction of the traces, despite the prefixes being allocated, they are not advertised in BGP (even in full RIBs). Table IV shows that more than 20% of the selected IPs disclosed the absence of a CP, with no matching BGP prefix. This effect is even worse in *uth*, that exhibits RIBs with slightly less entries than the other VPs.

VI. Results: the Mismatch Rates in the Wild

In this section we present the results derived from our measurement campaign. In Sec. VI-A we provide an overall view of the MM rate for the four bounds we propose. Then, our study focuses in the *Lower* model, that has the most conservative design and presents the lowest MM rate. First, we analyze the impact of the set of rules that compose its mapping relaxation stage, in Sec. VI-B. Finally, in Sec. VI-C we gather the VPs where the *Lower* Bound has the highest values and try to identify the type of MMs, or *lies*, that cause this outcome.



(a) MM Rate according to the model in use. *Restricted* and *Lower* bound differ in less than 5% for all VPs. The MM rate for the lower bound is more than 35% for *cle* and *hm1*, more than 7 times compared to what is seen in the remaining VPs.



(b) Ratio of matches in the *Lower* Bound that result from of extending the model of the *Upper* bound, including the mapping relaxation stage. In general, the `SIB` rather than the `looseTPA` rule proves to be more useful.

Fig. 3: Analysis for daily measurements carried between 05.10.2018 and 17.11.2018 for the VPs belonging to the Peering testbed. In the case of the homemade VPs, results are based on measurements carried out during approximately 8 months (from 18.04.2018 to 19.12.2018).

A. MM rates observed in the wild for all models

The MM rate ($\mu \pm \sigma$) for the proposed models of our framework, as seen from the peers introduced in Table IV, is shown in Fig. 3a. The results are consistent across the VPs, i.e. the bounds from *Lower* to *Raw* always report an increasing number of MMs. Yet, *quantitatively* distinct patterns among the different peers can be observed, specially for *cle* and *hm1*.

Analyzing Fig. 3a in more detail, we note that ***Restricted* and *Lower* bounds differ in less than 5% for all VPs**. Moreover, their values are lower than 5% in most cases. This small difference suggests that **TPAs and wildcards resulting from unresponsive hops and/or undefined mapping are either not frequently found in sequence or, when they are, the DP still matches the CP.** Recalling Sec. IV-A, this result shows that the *Lower* model does not gain much from implementing `looseTPA` rather than `strictTPA`, neither from the more conservative ordering of the rules applied in the mapping relaxation stage (`SIB` rule after `TPA` rule).

On the other hand, *Raw* and *Upper* bounds also perform similarly, though the latter generally shows a MM rate just a bit lower than the former. Therefore, in most cases, wildcards resulting from unresponsive hops and/or undefined mapping can be silently discarded. The only exception is *cle*, where the difference amounts to 23% due to an AS deletion that occurs at the beginning of many DPs. Also, note that in the *Lower* and *Restricted* models, `TPA` rules in the mapping relaxation stage exchange inferred TPAs for wildcards, thus increasing the need of the wildcard correction step.

According to the design of the proposed framework, the *real rate of lies* observed by VP is expected to be between their respective *Lower* and *Upper* bounds. In other words, the MMs observed only through the *Upper* model are potential *false negatives* for the *Lower* model, i.e. potential lies wrongly filtered as noise. Consequently, the rate of lies may be as significant as the *Upper* bound, at worst. On the other hand, its value could be closer to the fully-conservative *Lower* bound, usually less than 5%. While this value is low, *it is not negligible: according to its conservative design, the Lower bound is expected to filter most of the noise and to capture many actual lies.*

B. Effect of `SIB` and `TPA` rules on the MM rate

Our models can be grouped both in terms of design and performance: *Raw* and *Upper* on one side, and *Restricted* and *Lower* on the other. As illustrated in Fig 3a, there exists a large gap in terms of the MM rates seen for these two groups, except in *hm2* where all bounds are surprisingly close to each other. While *Raw* and *Upper* models just apply a wildcard correction, *Restricted* and *Lower* make extensive use of the mapping relaxation stage. We now analyze if any of the rules in this block is more effective to decrease the number of MMs, or if it is rather their combination that is required. Since each rule was designed to treat a specific limitation affecting DPs and CPs, this would also reveal if there is an outstanding kind of noise biasing severely the ground data. In particular, since the *Restricted* and *Lower* bounds perform similarly, we focus only on the *Lower* bound, and thus `SIB` and `looseTPA` rules.

The difference between red and green bars for each peer in Fig. 3a represents the amount of MMs observed via the *Upper* model and not via the *Lower* one. In other words, it is the share of cases that benefit from the mapping relaxation stage. We analyze which of these cases actually profit from applying (i) only the `SIB` rule, (ii) only the `looseTPA` rule, and (iii) both. As shown in Fig. 3b, less than 3% of the total cases across all VPs are filtered from the MMs using concurrently both `SIB` and `looseTPA` rules. Indeed, this small proportion indicates that, in general, **paths do not include simultaneously AS siblings and TPAs.** In addition, between 68% and 97% of the cases across all VPs (*cle* and *hm2* being the exception with less than 32% and 50%, respectively) require only using the `SIB` rule. Therefore, since this rule filters most MMs, we conclude that *the effect of AS siblings is the greatest interfering factor when comparing CPs and DPs*.

C. Looking closer at high MM rates

Although pinpointing and understanding the root causes of observed MMs —and defining whether they are deliberate or not— is challenging (see Section I), the high MM rates observed for the *Lower* bound in *cle* and *hm1* (together with their higher variability over time) encouraged us to further investigate these cases. Indeed *cle*'s provider sends traffic directly to the AS that is expected to be two AS-hops away, according to advertised CPs. While the presence of an *unintended lie* is a likely cause—also in line with the high variability observed—neither an *interested lie* nor *AS poisoning* can be discarded. On the other hand, a privileged view in *hm1* allows us to access the ground truth and to determine that most MMs seen in this peer originate in *technical limitations* in the infrastructure of its provider AS. Indeed, limited space in the forwarding tables results in FIB inconsistencies in their network. This, combined with the adoption of a persistent default route at the border router that connects to *hm1*, causes the traffic to exit the provider AS through a peering AS that is not necessarily the one included in CPs.

VII. Conclusion

Inter-domain *lies* are not straightforward to detect: noise in the ground data causing AS insertions, deletions, and substitutions can also generate MMs between CPs and DPs. Since this noise can be confused as lies—and vice versa—filtering it is imperative.

In this paper, we proposed a framework based on multiple path-rewriting rules that overcomes noise and allows to compute four MM-rate bounds to quantify lies. We leveraged the PEERING testbed that provides full-RIBs from multiple peers as well as co-located CPs and DPs, and carried out a longitudinal analysis as never done before, that spanned 8 VPs and up to 8 months of measurements. While the noise from TPAs and IXPs was more prevalent for a limited number of VPs, we observed that the noise due to AS siblings seems to generate most MMs.

Finally, we quantified the lower bound of the MM rate seen in the wild as being less than 5%. This value is small, but not negligible: since our approach is conservative, we expect to have filtered most of the noise and have captured many actual lies. Moreover, this also means that there might be many false negatives, i.e. many lies that finished being filtered as if they were noise. At the same time, we further analyzed the nature of MMs persisting after applying the most conservative filter in a VP where we have a privileged view, concluding that *technical limitations* in the infrastructure of the provider AS were causing them. Indeed, the combination of FIB inconsistencies and the extensive use of default routes are a potential cause for many MMs worth investigating.

We now aim at shedding light on the root cause of the MMs between CPs and DPs, i.e. we will focus in the even more challenging task of detecting not only *lies*, but also pinpointing their types and incentives, without counting on ground truth. Moreover, we plan on studying the feasibility of lies occurring both at the control and data planes simultaneously and on shielding our method against cases where `traceroute` is handled differently from regular traffic [6], [7].

Acknowledgments

We thank Randy Bush and Italo Cunha for their guidance setting up VPs and using the PEERING testbed, respectively. This work has been published under the framework of the IdEX Unistra and benefited from a funding from the state managed by the French National Research Agency as part of the "Investments for the future" program. This project has been made possible in part by a grant from the Cisco University Research Program Fund, an advised fund of Silicon Valley Foundation. This work has been partially funded by GRISIS project (CUP: B63D180002800079), DD MIUR prot.368 of 24/10/2018, Programma Operativo FESR Campania 2014–2020.

References

[1] Y. Hyun, A. Broido, and k. claffy, "Traceroute and BGP AS Path Incongruities," CAIDA, Tech. Rep., Mar 2003.
[2] Y. Zhang, R. V. Oliveira, Y. Wang, S. Su, B. Zhang, J. Bi, H. Zhang, and L. Zhang, "A framework to quantify the pitfalls of using traceroute in as-level topology measurement," *IEEE JSAC, 29, 1822-36, 2011*.
[3] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz, "Towards an accurate as-level traceroute tool," in *SIGCOMM 2003*. ACM, 2003, pp. 365–378.
[4] Z. M. Mao, D. Johnson, J. Rexford, J. Wang, and R. Katz, "Scalable and accurate identification of as-level forwarding paths," in *IEEE INFOCOM 2004*, vol. 3, March 2004, pp. 1605–1615 vol.3.
[5] S. Murphy, "BGP Security Vulnerabilities Analysis," Jan '06, RFC 4272.
[6] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev, "NetHide: Secure and Practical Network Topology Obfuscation," in *USENIX Security 2018*, August 2018.
[7] S. Trassare, R. Beverly, and D. Alderson, "A technique for network topology deception," in *Proc of the MILCOM*, Nov 2013.
[8] L. Gao and J. Rexford, "Stable internet routing without global coordination," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 681–692, Dec. '01.
[9] Y. Hyun, A. Broido, and k. claffy, "On Third-party Addresses in Traceroute Paths," in *PAM 2013*.
[10] P. Marchetta, W. de Donato, and A. Pescapè, "Detecting third-party addresses in traceroute traces with IP timestamp option," in *PAM 2013*.
[11] M. Luckie and k. claffy, "A second look at detecting third-party addresses in traceroute traces with the ip timestamp option," in *PAM 2014*, M. Faloutsos and A. Kuzmanovic, Eds.
[12] N. Ahmed and K. Sarac, "An experimental study on inter-domain routing dynamics using ip-level path traces," in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, Oct 2015, pp. 510–517.
[13] M. Luckie, A. Dhamdhere, B. Huffaker, D. Clark, and k. claffy, "bdrmap: Inference of Borders Between IP Networks," in *IMC 2016*, pp. 381–396.
[14] A. Marder and J. M. Smith, "MAP-IT: Multipass Accurate Passive Inferences from Traceroute," in *IMC 2016*. ACM, 2016, pp. 397–411.
[15] A. Marder, M. Luckie, A. Dhamdhere, B. Huffaker, k. claffy, and J. M. Smith, "Pushing the boundaries with bdrmapit: Mapping router ownership at internet scale," in *IMC 2018*. ACM, 2018, pp. 56–69.
[16] B. Schlinker, K. Zarifis, I. Cunha, N. Feamster, and E. Katz-Bassett, "PEERING: An AS for Us," in *HotNets-XIII*, 2014.
[17] X. Zhang and C. Phillips, "A survey on selective routing topology inference through active probing," *IEEE Communications Surveys Tutorials*, vol. 14, no. 4, pp. 1129–1141, Fourth 2012.
[18] P. Marchetta and A. Pescapè, "DRAGO: detecting, quantifying and locating hidden routers in traceroute IP paths," in *INFOCOM Workshops, Turin, Italy, April 14-19, 2013*, 2013, pp. 109–114.
[19] G. Nomikos and X. Dimitropoulos, "traixroute: Detecting ixps in traceroute paths," in *PAM 2016*, pp. 346–358.
[20] "PEERING testbed sessions," https://peering.usc.edu/peers/.
[21] M. Luckie, "Scamper: A scalable and extensible packet prober for active measurement of the internet," in *Proc. ACM SIGCOMM IMC*, Nov. 2010.
[22] "RIRs prefixes," https://labs.apnic.net/delegated-nro-extended, Apr. '18.