UNIVERSITÀ DEGLI STUDI DI ROMA TRE
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy

Update-Driven Root Cause Analysis in Interdomain Routing

ALESSIO CAMPISANO, LUCA CITTADINI, GIUSEPPE DI BATTISTA, TIZIANA REFICE, CLAUDIO SASSO

RT-DIA-117-2007 **June 2007**

(1) Dipartimento di Informatica e Automazione,
Università di Roma Tre,
Rome, Italy.
{gdb,ratm,refice }@dia.uniroma3.it
{alessio.campisano, claudio.sasso }@gmail.com

Work partially supported by European Commission – Fet Open project DELIS – Dynamically Evolving Large Scale Information Systems – Contract no 001907 and by MIUR under Project "MAINSTREAM: Algoritmi per strutture informative di grandi dimensioni e data streams", MIUR Programmi di ricerca di Rilevante Interesse Nazionale.

ABSTRACT

Interdomain routes change over time, and it is impressive to observe up to which extent. Routes, even the most stable, can change many times in the same day and sometimes in the same hour or minute. Such variations can be caused by several types of events, e.g., the change of the routing policies of an ISP, the reboot of a router, or the fault of a link. Some events are physiological to the network, while others are anomalous. In this paper we do a step towards the identifycation of the cause of route changes, a problem that is attracting increasing attention from both researchers and network administrators. Namely, we propose a methodology for analyzing a given BGP route change in order to, at least partially, identify and locate the event that triggered the change. Our data sources are the BGP updates collected by RIS and ORV projects. The methodology relies on a flow-based model of BGP updates. The cornerstones of the methodology are: data quality analysis that allows discarding unreliable data, analysis of the ranks of BGP peerings, and analysis of flows of prefixes. The methodology is supported by an on-line service.

1 Introduction

We consider the following scenario. A network administrator of an Internet Service Provider (ISP) observes that one of the prefixes announced by its Autonomous System (AS) to the Internet had a path change in the Border Gateway Protocol (BGP) interdomain routing at a certain time. For example, the prefix p announced by AS1 usually reaches AS4 passing through AS2 and AS3, while at a certain time it started using a different path through AS5 and AS6. The network administrator would like to know why that change happened.

One may object that the problem is not interesting, since there are so many changes in the interdomain routing and the Internet keeps working fine anyway. In contrast, recent papers (e.g., [24]) put in evidence the impact of routing changes in end-to-end performance. Also, this issue becomes much more important as services requiring almost constant delay, limited jitter and packet loss, grow in the Internet. Hence, the ISPs are more and more interested in understanding, as much as possible, what happens to their prefixes in the interdomain routing.

Actually, many research works have analyzed BGP routing dynamics in the last few years. The contributions can be broadly classified as follows.

There are black-box approaches, that apply statistical techniques to group BGP updates into sets that are supposed to be triggered by the same underlying event. To do that [27] uses the Principal Components Analysis, [22, 30] use statistics-based anomaly detection, and [29] exploits the wavelet transformations.

Other authors propose white-box approaches. In [9, 7, 6] streams of BGP updates are analyzed, correlating information across time, topology, collectors, and prefixes. [15] describes an algorithm, that pinpoints the origin of routing changes due to a link failure or a link restoration. This algorithm relies on a simplified model of BGP, assuming shortest path routing.

Finally, some authors (see, e.g., [21, 25]) propose to add an infrastructure to the Internet in order to monitor route changes. Even more, [17] proposes to modify the BGP protocol to facilitate the location of the causes of route changes.

However, after such a deep and intense research work, the problem of having a full understanding of the interdomain routing dynamics is far from being solved and the sentence of Griffin cited by many authors "in practice, BGP updates are perplexing and interpretation is very difficult" is still relevant.

As shown in the scenario described at the beginning of this section, we propose to tackle the problem from a new perspective. We assume the perspective of an ISP, that is not interested in what happens to the network in general but is rather interested in what happens to its own prefixes.

In this paper we present the following contributions.

- In Section 3, we show that BGP updates have a flow-based behavior, where the term flow is used with its graph-theoretic meaning. The collectors of updates are the sources of flow and the ASes originating prefixes are the sinks. Based on such observation, we propose a flow-based model of BGP updates. The model puts in a flow-based perspective the concept of link-rank, introduced in [13]. Also, the model presents the new concept of global-rank.

- In Section 5, we propose a methodology for analyzing a given BGP route change c in order to, at least partially, identify and locate the event that triggered c. The cornerstones of the methodology are: (i) A data quality analysis that allows to discard unreliable data using an approach similar to the one in [28]. (ii) A static flow analysis focused on local and global ranks. (iii) A dynamic flow analysis that examines prefix flow changes over time. Our data sources are described in Section 4. The methodology is supported by several examples from a reference week.

- In Section 6, we present an on-line service to support the methodology. The service relies on efficient algorithms for performing the computations described in the methodology and on an effective layout facility. Other services have been proposed to help understanding BGP updates. For example: (i) BGPlay [8] shows an animated graph of the as-paths used by traffic to reach a specific destination. (ii) LinkRank [13, 14, 2] shows the number of prefixes flowing through each BGP peering in Internet and provides animations of rank changes over time, showing how and when peerings loose and gain routes. (iii) BGP-Inspect [5, 1] allows users to easily access and extract information from raw BGP data. However, as far as we know, we present the first on-line service designed to support the analysis of one BGP route change.

2 Background

The Internet is divided into administrative domains called *Autonomous Systems* (*ASes*).

The *Border Gateway Protocol* (*BGP*) [18, 19] is the routing protocol used to exchange reachability information between ASes. Two ASes whose routers exchange routing information using BGP are said to have a *peering* between them.

A BGP router stores in its *Routing Information Base* (*RIB*) the *prefixes* it can reach, and for each of them an *AS-path*. An AS-path, also called *route*, is the sequence of ASes used to reach the destination prefix. Routes are propagated by BGP messages called *updates*.

BGP is an incremental protocol: once two BGP routers establish a peering, they exchange their whole RIB each other; this process is called *table transfer*. Further updates are sent only if a route changes, in response to network events (e.g., link failure, router reset, or policy change).

To obtain information about the evolution of the Internet routing state, projects such as the RIPE NCC's *Routing Information Service* (*RIS*) [4] and the University of Oregon's *RouteViews Project* (*RV*) [3], spread around the world several passive monitors, called *(Remote) Route Collectors* (*RRCs*). Each route collector peers with several BGP routers, called *Collector Peers* (*CPs*), belonging to various ASes. The routing tables of all RRCs and the updates they receive are periodically dumped, permanently stored, and made publicly available. Some collector peers provide information about all the prefixes on the Internet, while others only provide information about a subset of them. We call the former *full collector peers*, the latter *partial collector peers*.

3 A Flow-based Model

Several models have been proposed to study the evolution of interdomain routing. Most of them assume that each AS can be collapsed into a single router, while others [16] represent the internal structure of each AS with different levels of accuracy. The first approach is too abstract to capture the impact of the internal routing of an AS on the evolution of the Internet. On the other hand, the second approach contrasts with the fact that the currently available methodologies and data are not able to provide a fine-grained complete and accurate description of the internal structure of an AS.

In this section, we introduce a model based on the concept of flow. The model is shown to be valid independently on the internal structures of each AS. The validity of the model has the benefit of allowing correct deductions in Root Cause Analysis of interdomain routing; of course, it also has the drawback of not capturing dynamics internal to an AS.

We consider the following sets. $\mathcal{ASes}$ is the set of all the known ASes, $\mathcal{ASes} = [1, 65535]$. Since we will consider a graph whose nodes are the elements of $\mathcal{ASes}$, the ASes will also be called *vertices*, and pairs of ASes will be called *edges*. $\mathcal{Times}$ is the set of all the considered instants of time when a BGP update is received by a RRC from a collector peer; a total order $<$ is defined on $\mathcal{Times}$. $\mathcal{CollectorPeers}$ is the set of all the collector peer identifiers. Usually, each collector peer has exactly one peering session with one RRC. Hence, we can use its IP address as identifier. If a collector peer has more than one peering per RRC, we assign to it one distinct identifier for each peering session. $\mathcal{Prefixes}$ is the set of all known prefixes.

An *AS-path* (or simply *path*) π is a sequence of ASes such that $\pi = (as_n, as_{n-1}, \ldots, as_1, as_0)$ where $as_i \in \mathcal{ASes}$. as_0 is called *origin*. The empty path is an AS-path and is denoted by ϕ. $\mathcal{AsPaths}$ is the set of all known AS-paths. A pair (as_{i+1}, as_i) of ASes that are consecutive in some AS-path is an *edge*. We consider the edges as directed, i.e. $(v,w) \neq (w,v)$. We say that a path *contains* an edge, $\pi \supseteq (as_{i+1}, as_i)$. The relation *compatibility* ($\bowtie$) is defined on $\mathcal{AsPaths}$; two paths π_1 and π_2 are *compatible* ($\pi_1 \bowtie \pi_2$) when they share a common left subsequence of at least two ASes.

An *update* u is a quadruple (cp, p, π, t) where $u.cp \in \mathcal{CollectorPeers}$ is the CP that collected the update, $u.p \in \mathcal{Prefixes}$ is the prefix contained in the update, $u.\pi \in \mathcal{AsPaths}$ is the route announced by the update, and $u.t \in \mathcal{Times}$ is the time when the update is collected. If $u.\pi \neq \phi$ then u is an *announcement*, otherwise it is a *withdrawal*. $\mathcal{Updates}$ is the set of all known updates. $\mathcal{Updates} \subseteq \mathcal{CollectorPeers} \times \mathcal{Prefixes} \times \mathcal{AsPaths} \times \mathcal{Times}$. A partial order is defined on $\mathcal{Updates}$ as follows: $u_1 < u_2 \iff u_1.cp = u_2.cp \wedge u_1.t < u_2.t$.

The *last* update u that collector peer cp received for prefix p, before time t, is denoted $\ell_{cp}(p,t)$; formally, $\ell_{cp}(p,t)$ is such that $\ell_{cp}(p,t).t < t$ and $\nexists u \in \mathcal{Updates} \mid u.cp = cp \wedge u.p = p \wedge \ell_{cp}(p,t).t < u.t < t$.

Two consecutive updates u and $\ell_{u.cp}(u.p, u.t)$ cause a routing *change*; a change c is a quintuple $(cp, p, \pi_{old}, \pi_{new}, t)$ where $\pi_{old} = \ell_{cp}(p,t).\pi$ and $\pi_{new} = u.\pi$

We now define two concepts that will be crucial for the methodology described in Section 5, called local rank and global rank. While the former has been introduced in [13], the latter is, as far as we know, an unexplored concept. Given a collector peer cp, we define the *local rank* of an edge e at time t as the number of prefixes whose path at time t, as observed by cp, contains e. Namely, $lrank(cp, e, t) = |P_{cp}(t)|$, where $P_{cp}(t) = \{p \in \mathcal{Prefixes} \mid e \subseteq \ell_{cp}(p,t).\pi \vee \exists u = (cp, p, \pi, t) \in \mathcal{Updates} \mid u.cp = cp, u.t = t, e \subseteq u.\pi\}$. We define the *global rank* of an edge e at

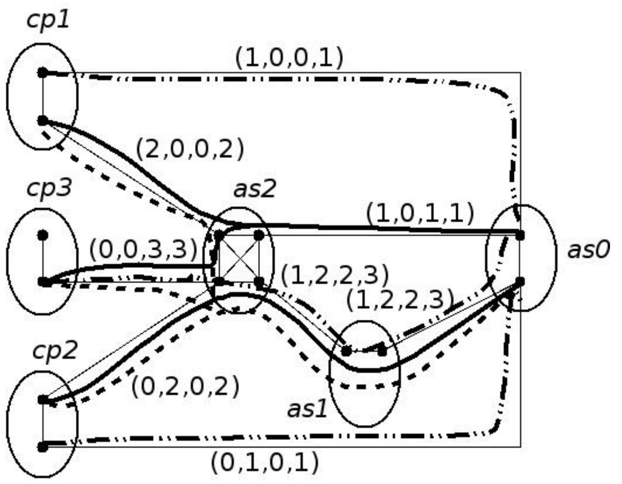


Figure 1: Points represent routers, thin solid lines represent IBGP or EBGP peerings between routers, and ellipses represent ASes. ASes cp_i, $i \in \{1,2,3\}$, contain collector peers. Each edge e inter-AS is labelled with $lrank(cp_1,e,t)$, $lrank(cp_2,e,t)$, $lrank(cp_3,e,t)$, and $grank(e,t)$. AS as_0 originates three prefixes. Dotted, dashed, and mixed lines represent the routes to such prefixes observed by collector peers.

time t as

$$grank(e,t) = |P(t)|, P(t) = \bigcup_{cp \in \mathcal{CollectorPeers}} P_{cp}(t).$$

Intuitively, while the local rank measures the number of prefixes that are observed passing through an edge by a single cp, the global rank measures the number of distinct prefixes that are observed passing through an edge by any $cp \in \mathcal{CollectorPeers}$.

Figure 1 illustrates the values of local and global ranks of the edges of a fragment of Internet, at a certain time t.

Besides these functions, the need for some static features to characterize edges in such a dynamic environment, leads to the definition of the following average values: the weighted average of the lrank of an edge in a time interval $[t_0,t_n]$, where instants t_i are such that $lrank(cp,e,t)$ is constant in $[t_{i-1},t_i)$

$$\overline{lrank}(cp,e) = \frac{\sum_{i=1}^{n} lrank(cp,e,t_i)(t_i - t_{i-1})}{t_n - t_0}$$

and, similarly, the weighted average of the grank of an edge in the same time interval

$$\overline{grank}(e) = \frac{\sum_{i=1}^{n} grank(e,t_i)(t_i - t_{i-1})}{t_n - t_0}$$

Whenever a negative (positive) variation of a local rank is observed during a given time interval, it is interesting to further investigate where prefixes “went to” (“came from”). Intuitively, prefixes move around on the AS graph, as well as water would move in a pipe network. This analogy introduces the concept of flows of prefixes. Tracking the flow of prefixes along different paths can be performed by relating the classical flow model to the AS graph. Hence, we adapt the classical concept of flow system to the interdomain routing domain. In fact, since each AS originates its own prefixes, we are considering a single source/multiple sink network flow model; besides this, we are not constrained by edge capacity. Namely, given a graph $G = (V,E)$, a specific vertex as_n called *source*, and a mapping between vertices and flow absorption $g : V \longrightarrow \mathbb{Z}$, then a *flow system* is a function $f : V \times V \longrightarrow \mathbb{Z}$ where $\forall v \in V, v \neq as_n$,

$$\sum_{w \in V} f(w,v) - \sum_{w \in V} f(v,w) = g(v).$$

Theorem 1 shows that functions $lrank(cp,e,t)$ defined above, and $\theta(cp,v,t) = |P(cp,v,t)|$, where $P(cp,v,t) = \{p \in \mathcal{Prefixes} \mid \ell_{cp}(p,t).\pi = (as_n,\ldots,v), n \geq 0 \ \vee \ \exists u \in \mathcal{Updates} \ \mid \ u.cp = cp, u.t = t, u.\pi = (as_n,\ldots,v)\}$, define a flow system at time t. Notice that function $\theta(cp,v,t)$ is the number of prefixes that, at time t, are known by cp as originated by each AS v.

Theorem 1. *Functions $lrank(cp,e,t)$ and $\theta(cp,v,t)$ define a flow system at time t.*

Proof. Select a specific t and a specific cp. Consider the value $x = \theta(cp,v,t)$ of function θ for vertex v. Because of the definition of θ we have that for each unit of flow in x there exists a prefix p such that either $p \in \mathcal{Prefixes} \mid \ell_{cp}(p,t).\pi = (as_n,...,v), n \geq 0$ or $\exists u \in \mathcal{Updates} \mid u.cp = cp, u.t = t, u.\pi = (as_n,...,v)$. In both cases we identify an update u that is received from as_n and originates from v. Consider a sequence of two consecutive edges (as_{i+1},as_i) and (as_i,as_{i-1}) contained in $u.\pi$: u contributes with one unit of flow both to $lrank(cp,(as_{i+1},as_i),t)$ and to $lrank(cp,(as_i,as_{i-1}),t)$. Hence, for each AS $as_i \neq v$, u does not affect the balance of as_i. This means that for each vertex v, if we consider only paths not ending with v we have

$$\sum_{w \in V} lrank(cp,(w,v),t) = \sum_{w \in V} lrank(cp,(v,w),t).$$

Now, each path ending with v increases both the flow on an incoming edge, $lrank(cp,(w,v),t)$, and $\theta(cp,v,t)$. Then we conclude that, $\forall v \in V$,

$$\sum_{w \in V} lrank(cp,(w,v),t) = \sum_{w \in V} lrank(cp,(v,w),t) + \theta(cp,v,t).$$

□

Intuitively, we have that the source of the flow is the AS as_n in which the selected collector peer cp is located, and the sinks are all the other ASes that originate some prefixes, as observed by cp. As an example, consider collector peer cp_1 in Figure 1. $\theta(cp_1,as_0,t) = 3$ and $\theta(cp_1,as,t) = 0 \ \forall as \neq as_0$. If we consider a collector peer, say cp_1, and any AS, for instance as_2, we have that the sum of the local ranks over incoming edges $(cp_1,as2)$ is 2, and the sum of the local ranks over outgoing edges (as_2,as_0) and (as_2,as_1) is 2.

On the other hand, notice that functions *grank* and θ do not define a flow system. As a counterexample, consider again AS as_2 of Figure 1. The algebraic sum of the global ranks of the edges incident on as_2 is not zero.

Theorem 1 is useful to depict a snapshot of the network at a given instant. In Theorem 2, we relate the flows of two different instants of time. We define the functions

$$\Delta lrank_t^{t+\tau}(cp,e) = lrank(cp,e,t+\tau) - lrank(cp,e,t)$$

that captures local rank variations (flow variations) between t and $t+\tau$, and the function

$$\Delta\theta_t^{t+\tau}(cp,v) = \theta(cp,v,t+\tau) - \theta(cp,v,t).$$

that accounts for the variation in the number of prefixes that are known by cp as originated by v.

Theorem 2. *Functions $\Delta lrank_t^{t+\tau}(cp,(v,w))$ and $\Delta\theta_t^{t+\tau}(cp,v)$ define a flow system.*

Proof. For the sake of simplicity, we use $l((v,w),t)$ in substitution of $lrank(cp,(v,w),t)$. $\forall v \in V$:

$$\begin{aligned}
&\sum_{w \in V} \Delta lrank_t^{t+\tau}(cp,(w,v)) - \sum_{w \in V} \Delta lrank_t^{t+\tau}(cp,(v,w)) = \\
&\sum_{w \in V} l((w,v),t+\tau) - \sum_{w \in V} l((v,w),t+\tau) - \sum_{w \in V} l((w,v),t) + \sum_{w \in V} l((v,w),t) = \\
&\theta(cp,v,t+\tau) - \theta(cp,v,t) = \\
&\Delta\theta_t^{t+\tau}(cp,v).
\end{aligned}$$

□

Observe that, because of the high connectivity of the Internet, we expect that function $\Delta\theta_t^{t+\tau}(cp,v)$ is zero in most cases. That is, we expect that the flow is overall conserved over time.

Project	CPs	full CPs
RIS	395	84
RV	101	85

Table 1: Statistics on CPs.

4 Data Set

Our work relies on BGP RIB dumps and updates obtained from RIS [4] and RV [3]. The data we use throughout this paper have been collected in the week from 12/26/2006 to 01/02/2007. At that time, there were 496 collector peers; 30% of them were full collector peers (see Table 1 for details).

Our dataset contains 320,678,893 updates (nearly 46M updates per day on average) with 7,537,378 distinct AS-paths on 70,078 distinct peerings and 24,493 distinct ASes; the number of observed prefixes is 235,725.

Route Collectors Reliability To check the reliability of route collectors, we periodically perform on all RRCs a preprocessing step, called *Route Collector Reliability Screening*.

The Reliability Screening on a route collector rrc' in a particular time interval $[t_{start}, t_{end}]$ is executed in the following way: starting from the RIB of rrc' at t_{start}, we make a local copy of that RIB, and we modify the copy according to the updates collected by rrc' during $[t_{start}, t_{end}]$; then we compare the modified copy to the RIB dumped by rrc' at t_{end}; we decide if rrc' is unreliable by evaluating the ratio between the number of inconsistencies found and the average size of the RIB.

Possible causes of the unreliability of a route collector are: bugs in the implementation of routing or collection software (some of which have been studied in [11]), major asynchronies between the route collector and the collector peer, non-standard behavior of the collector peer (e.g. some highly recommended timers are not implemented).

Reliability Screenings performed during several experiments led to the detection of a major problem that affected RIS route collectors since May 2005; overall, the problem affected 12 full collector peers. Contacting the RIS maintainers resulted in that problem being fixed by January 2nd, 2007. Since we chose the reference week before the problem was solved, we are thus able to assess the impact of this data cleaning step by comparing computations executed on “dirty” and “clean” data.

5 A Methodology for Analyzing a Route Change

We present a methodology for analyzing a given route change c within the model described in Section 3, where c could as well be an announcement or a withdrawal. The target is to identify the portion of the Internet where the event that caused c happened. This is done by analyzing the available data across several dimensions and perspectives. The methodology consists of three steps.

- *Collector Peer Check and Location*: First, we check the reliability of all the collector peers at the time $c.t$ of the route change, and select a set of collector peers that will be the most used in the analysis.
- *Static Flow Analysis*: In this step we look for patterns of macro-events. This is done exploiting the global and local ranks of edges, relying on Theorem 1.
- *Dynamic Flow Analysis*: In this step we deepen the analysis, relying on Theorem 2. Namely, if no macro-event has been detected in the previous step, we perform a fine-grained analysis based on several patterns that are consequences of the theorem.

Before entering the description of the steps, we have to discuss an issue related to events timing. In several points of the methodology, we analyze what happens in a time interval including the time $c.t$ of the path change. The literature suggests that a reasonable compromise between accuracy and feasibility for choosing a time interval, centered in $c.t$, is 180 seconds [12]. However, the methodology is independent on this choice, hence in the following sections, for the sake of simplicity, we refer to that time interval with informal terms like “near”,“close to” or “around”.

5.1 Collector Peer Check and Location

Collector Peer Check The methodology assumes (Section 4) that a route collector reliability screening has been performed before analyzing any route change. This activity discards all the route collectors that are considered not reliable. However, after that we have still to check if the available data can be considered reliable for the specific analysis of c.

More precisely, in [28] it has been shown that collector peers are frequently subject to reboot. It is clear that a reboot of the collector peer $c.cp$, that observed c, around time $c.t$ would cause too much noise on the available data to continue the analysis. Also, in the analysis of c, we will rely not only on $c.cp$, but also on other collector peers. Hence, in this step we identify all the collector peers that had a reboot that happened near $c.t$. We check the occurrence of a reboot by either analyzing BGP session state messages, when available, or by seeking for table transfers using the algorithm described in Section 6. Information extracted from those collector peers is not further considered.

In order to estimate the effectiveness and the usefulness of the Collector Peer Check step, we analyzed the collector peer reliability during the reference week, taking into account only route collectors that successfully passed the preliminary screening.

We have that, considering only the 169 full collector peers that are the most relevant source of information, Algorithm 1 identified 90 table transfers, each one corresponding to a reset. BGP session state messages, only available for RIS collectors, reported 71 session resets of such collector peers. Overall, the percentage of time affected by session resets has been on average 3% of the reference week per collector peer. This seems to indicate that the available data is quite unreliable, however we have that 3 collector peers spent more than 10% of the time rebooting. Such collectors are the main responsible for these data and they are not full collector peers.

Peer Location Among all the available collector peers we locate those that belong to the ASes of the paths $c.\pi_{old}$ and $c.\pi_{new}$. Those collector peers are the most relevant for the activities performed in the subsequent steps of the methodology.

5.2 Static Flow Analysis

In this step we look for the possible presence of macro-events by performing first a global rank analysis and then a local rank analysis.

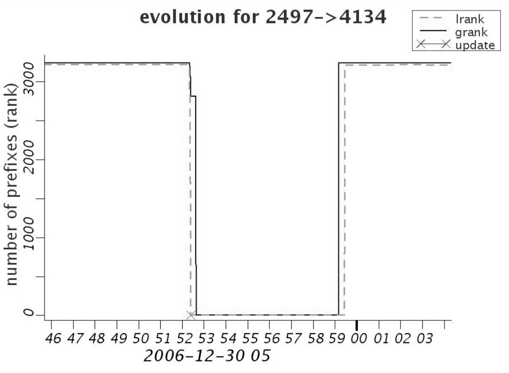
Global Rank Analysis The evolution of the global rank $grank(e,t')$ with t' near $c.t$ is considered for each edge e in $c.\pi_{old}$ and $c.\pi_{new}$. Namely, we check if some edge e in $c.\pi_{old}$ or in $c.\pi_{new}$ had a relevant global rank variation and reached a value near to zero within a time close to $c.t$. This would mean that all those prefixes that were seen passing through e by some collector peers are no longer flowing through e. This is a reasonable evidence that e is involved in some way in the event that caused c. There are at least three different patterns to consider for the global rank: (i) a sudden loss of all prefixes, (ii) a sudden gain of new prefixes starting from 0, or (iii) a sudden loss (gain) followed by the resume of the previous situation. Depending on the pattern we could be in presence of a link failure, a link restoration, or a BGP router reset.

The global rank of an edge is more or less representative of the actual state of a link depending on its value of *rank diversity*. We define the rank diversity of e as a pair $\langle n, \sigma_x/\bar{x}\rangle$, where n is the number of collector peers cp having $\overline{lrank}(cp,e) > 0$, σ_x and $\bar{x}$ are the standard deviation and the average, respectively, of such $\overline{lrank}(cp,e)$. The rank diversity is high if n is large and $\sigma_x/\bar{x}$ is small. In fact, if n is large we have many collector peers that can "see" e, and when $\sigma_x/\bar{x}$ is small we have that the collector peers see a comparable number of prefixes through e. Hence, we think that the global rank analysis provides valuable information only if the considered edges have a high rank diversity.

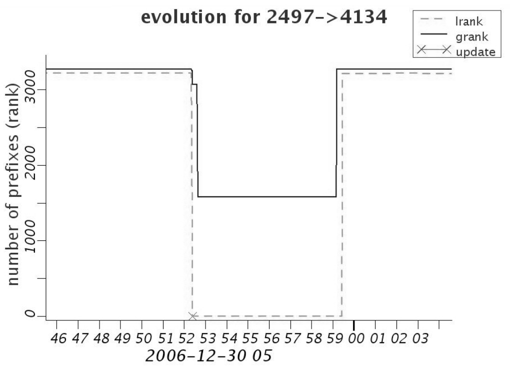
As an example, let us analyze the path change c, where $c.p = 202.41.242.0/24$, $c.cp = 198.32.176.24$ (the route collector is $rrc : routeviews - isc$), $\pi_{old} = (2497, 4134, 4847, 37942)$, $\pi_{new} = (2497,2914,4134,4847,37942)$, and $c.t =$ UTC 30/12/06 05:52:24.

First, we check collector peers reliability. Although we identified ~ 20 collector peer resets near $c.t$, they affected only collector peers having RIB size less than 100 prefixes. Hence, we can continue the analysis.

Then we evaluate the global rank of all edges belonging to $c.\pi_{old}$ and $c.\pi_{new}$. $grank(e) = 0$, near $c.t$, where $e = (2497, 4134)$. Looking at the average grank, we have $\overline{grank(e)} = 3166$, thus we have found a significant rank variation. Figure 2.a shows the evolution of $grank$ and $lrank$ (observed by $c.cp$) of e, near $c.t$.



(a)



(b)

Figure 2: Functions *grank* (solid black) and *lrank* (dashed gray) of $(2497,4134)$. (a) After collector peer screening. (b) Before collector peer screening.

	all cps	**w/o unreliable cps**
all edges	6332	8528 (+34,68%)
only edges with $\overline{grank}(e) > 50$	372	455 (+22,31%)
only edges with $\langle n \geq 3, \sigma_x/\bar{x} \leq 30\% \rangle$	13	16 (+23,08%)
only edges with $\overline{grank}(e) > 50$ **&** $\langle n \geq 3, \sigma_x/\bar{x} \leq 30\% \rangle$	7	9 (+28,57%)

Table 2: Number of distinct edges e with $\overline{grank(e)} > 0$ and $grank(e,t) = 0$ for at least one t in the reference week.

Since the rank diversity of e is very high, $\langle 7, 8.2\% \rangle$, we consider its global rank valuable. From the information extracted, we can argue with reasonable confidence that the path change c has been triggered by a major event, e.g. a link fault, lasting about 7 minutes on edge $(2497,4134)$.

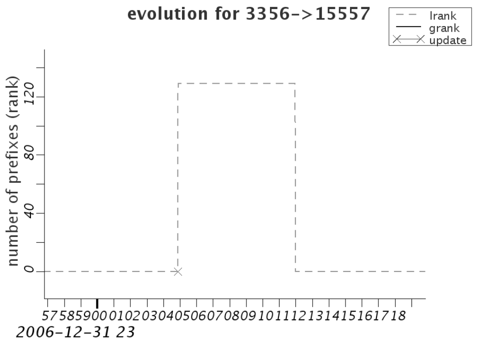
As a side note, computing *grank* without excluding collectors which didn't pass the screening (Figure 2.b), results in a *grank* that never decreases beyond 80. In this case, we would have never been able to spot the fault location through Global Rank Analysis. This highlights the importance of our efforts to clean data from the very beginning of the methodology.

To estimate the inference power of this step, Table 2 shows the number of edges having at least one instant t with $grank(e,t) = 0$. Such edges are further classified according to their average grank and their rank diversity. The values we have obtained are unexpectedly high, showing that a complete loss of BGP peering is an event with non negligible frequency. Comparing the last two columns of the table we also observe how the elimination of unreliable collector peers through the screening step is fundamental for locating major events.

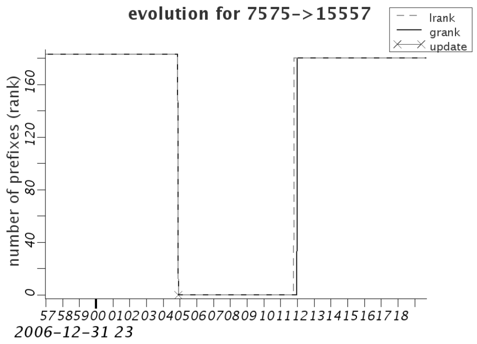
Local Rank Analysis If the Global Rank Analysis fails, then the evolution of the Local Rank $lrank(cp,e,t')$ with t' near $c.t$ is considered for each edge e in $c.\pi_{old}$ and $c.\pi_{new}$ and for each relevant collector peer cp selected in the first step of the methodology. We check those edges for the same aforementioned patterns, ensuring that such patterns are observed by the collector peers able to reveal them (i.e., all those located on the left side of the edge in $c.\pi_{old}$ or in $c.\pi_{new}$). Generally, we trust global ranks more than local ranks, so any inference supported only by local rank analysis requires further investigation. The only situation when we prefer the deduction supported only by local ranks is when the collector peer belongs to the left AS in the edge and provides its full routing table.

As an example, let us analyze the path change c, where $c.p = 80.124.192.0/19$, $c.cp = 198.32.176.177$ (the route collector is $rrc : routeviews - isc$), $\pi_{old} = (7575, 15557, 8228)$, $\pi_{new} = (7575, 2914, 3356, 15557, 8228)$, and $c.t =$ UTC 01/01/07 00:04:53.

The collector peers reliability check reveals no reboot of collector peers occurred near $c.t$, so all available collector peers will be considered. We evaluate the global rank of all edges belonging to $c.\pi_{old}$ and $c.\pi_{new}$, and we find out that $grank(e) = 0$, near $c.t$, where $e = (7575,15557)$. Note that $\overline{grank(e)} = 148$. Unlike the previous example, the rank diversity of e is $\langle 2, 0.1\% \rangle$, so its global rank is not worth consideration as the edge is seen by only two collector peers. Furthermore, such collector peers are both located in the same AS7575. As a consequence, we analyze *lrank* for $c.cp$. Being $c.cp$ in the left node of e, it is in the best position to observe routing events happening in e. Figure 3 illustrates the evolution of local ranks, as seen by collector peer $c.cp = 198.32.176.177$, and global ranks for edges (7575,15557) and (3356,15557). It is interesting to notice that a relevant number of prefixes moves from an edge to the other. From the information extracted, we can deduce with reasonable confidence that the path change c has been triggered by some event, occurred on the edge e, whose impact on the Internet has been quite limited, since no other collector peer was able to sense its effects.



(a)



(b)

Figure 3: Functions *grank* (solid black) and *lrank* (dashed gray) of $(3356, 15557)$ (a), and $(7575, 15557)$ (b).

5.3 Dynamic Flow Analysis

As we have seen in Section 3, flow changes are a powerful tool to analyze the network evolution. Sometimes, however, we're interested in monitoring just a portion of the flow changes. We therefore introduce *restricted flows*. A restricted flow $\Delta_t^{t+\tau}\hat{f}_P(cp,(u,w))$ is defined on a subset $P \subseteq \mathcal{Prefixes}$ of the prefixes.

Same-Origin Route Changes Analysis The ISP of AS as_0 that experienced change c, is most probably interested in knowing if its other prefixes, besides $c.p$, also had a path change near $c.t$. For this reason, we check if other prefixes, originated by the same AS as_0 that originated $c.p$, had a path change observed by the same collector peer $c.cp$, at a time close to $c.t$. Hence, we study the restricted flow corresponding to $P = \{p \mid \ell_{cp}(p,t+\tau).\pi = (as_n,\dots,as_0)\}$.

Namely, all changes affecting prefixes of as_0 are considered, labelling *stable* each prefix p_s whose path, as seen by $c.cp$, didn't change around $c.t$, and *floating* each prefix p_f whose path, as seen by $c.cp$, had a change around $c.t$. This leads to classifying each path, from the AS of $c.cp$ to as_0, as *stable*, if it is used by at least one stable prefix, or as *floating* otherwise. We consider each distinct change c_f of each floating prefix. Let Old be the set of the distinct paths π_o such that $c_f.\pi_{old} = \pi_o$ and let New be the set of the distinct paths π_n such that $c_f.\pi_{new} = \pi_n$.

Intuitively, if $|Old| \ll |New|$ ($|Old| \gg |New|$), then the cause is probably related to some path in Old (New), and we further investigate only this set. Moreover, if some stable paths are found in the Old (New) set, then we focus the analysis only on the New (Old) set. In fact, if stable paths that are found in the Old set do not change, then there is at least one path that remains available; it is more likely, thus, that the cause of the change is related to ASes in the New paths. The opposite case brings to symmetric considerations.

Otherwise, if none of the above patterns is found, a deeper analysis is required on both sets. One particular possibility is that $|Old| = |New| = 1$: in this case we have that all the prefixes of the origin ISP behave the same, and this is itself a valuable information for the ISP as_0.

As an example, we analyze the path change c, where $c.p = 202.59.174.0/24$, $c.cp = 80.81.192.143$ (the route collector is $rrc : ris - rrc12$), $\pi_{old} = (16215, 3549, 5511, 4761, 17727)$, $\pi_{new} = (16215, 3549, 3320, 4761, 17727)$, and $c.t =$ UTC 12/27/06 10:06:17.

After carrying out the aforementioned steps, we were not able to extract relevant information neither from the global nor from the local rank analysis. Hence, we analyze how all other prefixes originated by AS17727 behaved around $c.t$: we had 1 stable prefix and 22 floating prefixes. Path (16215, 3549, 7473, 4761, 17727) resulted the only stable path. Clustering the discovered path changes, we observe

- 4 prefixes changing from (16215, 3549, 5511, 4761, 17727) to (16215, 174, 5511, 4761, 17727)
- 4 prefixes changing from (16215, 3549, 5511, 4761, 17727) to (16215, 3549, 7473, 4761, 17727)
- 14 prefixes changing from (16215, 3549, 5511, 4761, 17727) to (16215, 3549, 3320, 4761, 17727)

Notice that, in this case, $|Old| = 1$ and $|New| = 3$; moreover, the New set contains a path that already existed when c happened. Figure 4 helps visualizing the currently studied path changes. Since multiple events happening

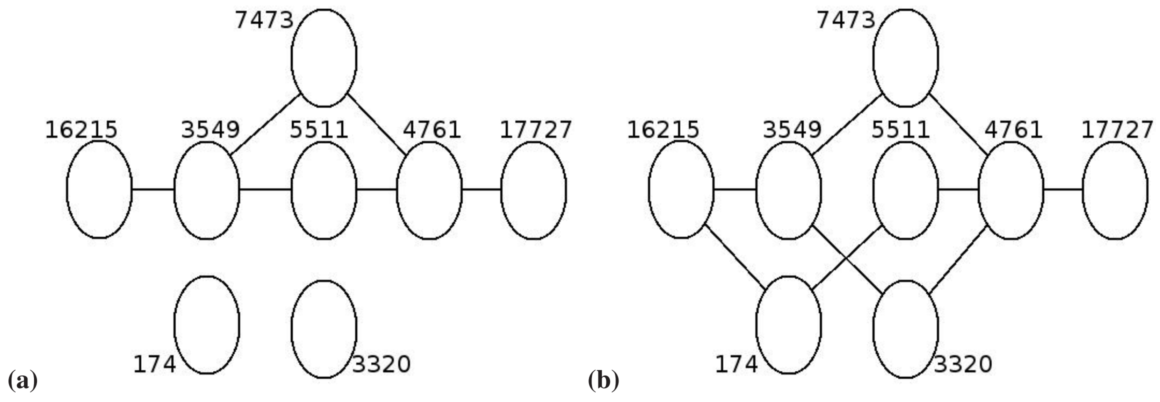


Figure 4: Paths to reach prefixes of AS17727 from the collector peer. (a) Before the path changes. (b) After the path changes.

on the three *New* paths are less likely than a single event happening on the single *Old* path, and since *New* also contains a stable path, we decide to further explore (16215, 3549, 5511, 4761, 17727).

To have an idea of the impact of this step of the methodology, it is interesting to examine the results presented in [16]. That paper studies the distribution of the number of distinct AS-paths observed in the Internet between each AS pair. Such distribution puts in evidence that this kind of "path diversity" happens with high frequency.

Cross Collector Peer Analysis We evaluate the set of the stable paths, as defined in the step above, for each collector peer in each AS in $c.\pi_{old}$ and $c.\pi_{new}$. Every edge belonging to a stable path is no more considered to be a candidate responsible of the routing change under analysis. Intuitively, we are checking whether the path changes for origin AS as_0 observed by $c.cp$ correspond to analogous changes observed by other collector peers along the path. In fact, being BGP an incremental protocol, no updates will be sent from an AS for its stable prefixes. If we consider the set of the collector peers belonging to an AS to correctly represent the AS's point of view, this means that the AS of the cp didn't send any update, and so the cause for the routing change has to be searched for elsewhere. Depending on how many collector peers we have on a single path, this activity can lead to the identification of a reasonably small set of candidates.

Resuming the analysis of the change c from the previous step, we move onto other collector peers, "walking" along our $c.\pi_{old}$ from AS16215 towards AS17727. AS3549 contains three collector peers. After computing stable and floating paths, they all agree in showing that subpaths (3549, 7473, 4761, 17727) and (3549, 3320, 4761, 17727) were already available and stable, while subpath (3549, 5511, 4761, 17727) was floating. This strengthens our guess about the cause of the event being related to the old path, and probably this tells us that AS16215 cannot be considered responsible for c since changes are being simply propagated from AS3549. Carrying out the same activity on the three collector peers available at AS5511 shows that, near $c.t$, the subpath (5511, 4761, 17727) was clearly floating. This leads to excluding AS3549 from the alleged causes of c. The step ends here since no collector peer is available at AS4761. Collector peers belonging to AS17727, if any, would be disregarded anyway, as they can only provide trivial information. Summarizing, at this point we have that the candidates for hosting the event that caused c are (5511, 4761, 17727).

To get a rough idea of the potential impact of this step, consider that our 496 collector peers are hosted by 314 distinct ASes. Hence, the probability to find a collector peer in a randomly chosen AS is about 1.3%. However, if we compute, using the algorithm proposed in [16], the set of tier-1 ASes, we obtain a collection of 11 ASes (very similar to the one shown in that paper). Only 3 of such ASes result in not having any collector peer. This is especially important since most AS-paths (949 out of 1000 distinct randomly chosen paths) in our dataset contain at least one tier-1 AS. Observe that this is interesting by itself, since it underlines that the vast majority of our updates do not contain peer-to-peer relationships in intermediate levels of the hierarchy.

Flow Changes Analysis A particularly interesting restricted flow is the one defined on those prefixes that experienced, in $[t, t+\tau]$, a change in which either the old or the new path is compatible with a given path π. That is, $P = \{p \mid \ell_{cp}(p,t).\pi \neq \ell_{cp}(p,t+\tau).\pi \wedge (\ell_{cp}(p,t+\tau).\pi \bowtie \pi \vee \ell_{cp}(p,t).\pi \bowtie \pi)\}$. Namely, we evaluate $\Delta_t^{t+\tau}\hat{f}_P(cp,(u,w))$. This can help understand if the cause of the path change is located on the new path, on the old one, or even on another, completely different, path. Notice that this tackles a common limitation of inference

# of prefixes	New path
+744	16215 3549 **3320 4761**
+119	16215 3549 **7473 4761**
+75	16215 3549 **3491 4761**
+2	16215 3549 4637 **3491 4761**
-944	16215 3549 **5511 4761**
-3	**4761 4795**
-1	**4761 23679**

Table 3: Relevant portion of $\Delta_t^{t+\tau}\hat{f}_P$. The table shows the edges (in **bold**) incident on 4761, the path they were extracted from, and their $\Delta_t^{t+\tau}\hat{f}_P(u.cp,e)$. Edges having $\Delta_t^{t+\tau}\hat{f}_P(u.cp,e) < 0$ belong to path changes that *bypass* AS4761, decreasing the overall number of prefixes passing through it.

systems based on the AS-path, i.e. the capability to locate causes of the change only on the new or the old path. Literature often refers to this problem as the "induced updates" problem [9].

We look for edges that experienced a significant flow change. Now, this can happen in two ways: a single edge e_1 can lose (gain) prefixes moving on (coming from) another single edge e_2, or a number of edges $e_3,\dots,e_k$. In the latter case, it's more likely that edge e caused the flow dispersion; in fact, multiple events affecting $e_3,\dots,e_k$ have a very low probability to happen at the same time. In the former situation, it is not clear which edge is more likely to be the cause of the change, so we deepen the analysis on e_2 studying other flow changes that involved e_2. This shift of focus makes our methodology inherently iterative, and allows to cope with the "induced updates" problem.

As an example, we advance our analysis of c from the previous steps, by studying restricted flows, i.e. $\Delta_t^{t+\tau}\hat{f}_P$ where the path π that defines the set of prefixes P is $c.\pi_{old}$ = (16215, 3549, 5511, 4761, 17727), and times t and τ are chosen in order to include $c.t$. Exploiting the insight of the cross collector peer analysis, we want to focus our attention mainly on (5511, 4761, 17727). We discover that $\Delta_t^{t+\tau}\hat{f}_P(u.cp,(4761,17727)) = 0$, i.e., flow doesn't change for that edge; by contrast, $\Delta_t^{t+\tau}\hat{f}_P(u.cp,(5511,4761)) = -944$, and $\Delta\theta_t^{t+\tau}(u.cp,4761) = 0$. This means that 944 prefixes left edge (5511,4761) switching to other edges, i.e. no prefix "seen" by $c.cp$ on that edge was announced nor withdrawn. Those 944 prefixes changed their path as shown in Table 3. It is important to underline that this kind of information cannot be extracted from the Local Rank Analysis alone.

We observe that one AS, namely 3549, spreads a relevant number of prefixes in multiple directions. On the other hand, 4761 gathers most of those prefixes from different edges. Thus, we conclude that the subpath (3549, 5511, 4761) contains the alledged cause of c. Merging this information with the output from all previous steps, we are able to further restrict our candidate set to (5511, 4761), since the Cross Collector Peer Analysis already excluded 3549.

The potential applicability of this step of the methodology is essentially the same as the other steps grouped in the Dynamic Flow Analysis.

6 A BGP-update analysis service

The methodology described in Section 5 requires the analysis of huge amount of data, and hence it would be unfeasible if not supported by some automatic facility. We developed an on-line service that offers many tools to support the methodology. A prototype version is available at `http://nerodavola.dia.uniroma3.it/rca/`.

The main building blocks of the service are:

- A crawler that periodically downloads updates from the data sources described in Section 4.
- A data cleaning system that checks the reliability of data sources, and identifies reboots of collector peers.
- A ranking system that computes local and global ranks.
- A tool that, on-demand, maps local rank variations to prefix flow changes.
- A GUI equipped with an automatic layout facility that shows information about a route change.

Table 4 shows technical specifications of the testbed used in the performance evaluation experiments reported in the current section.

CPU	2 x Intel Xeon 2.80GHz
Cache size	512 KB
Cached read time	594.16 MB/sec
Buffered disk read time	58.54 MB/sec
RAM size	3.8GB (User space 3.2GB)
OS	GNU/Linux

Table 4: Technical specifications of the testbed

6.1 Identification of Table Transfers

In order to maintain a clean dataset, we want to spot session resets between any collector peer and the route collector. If the session state messages of the collector peer are available, each transition from/to the BGP state "6=Established" [19] can be used to identify session resets; this is typically true for all RIS collector peers. For any other collector peer, this boils down to identifying, with a reasonable accuracy, all the BGP table transfer occurred between any collector peer and its route collector in a given collection of BGP updates. The identification of table transfers is useful even if we have the session state messages, since we need an estimate of the table transfer duration.

Although this issue had already been studied and solved in a very elegant way in [28], we found out that such an approach doesn't scale well over our large set of collector peers. In [28] no computational complexity analysis is given, however, it is not difficult to find that the proposed algorithm requires $O(n\omega\sigma)$ time and $\Omega(\frac{n}{\sigma})$ space, where n is the number of processed updates, σ is the maximum number of updates per second, and ω is the number of seconds of a window that is used to scan the updates (whose maximum size spans over two hours). From the point of view of the time complexity this is still acceptable if $\omega\sigma$ is $o(n)$. However, the requested main memory is by far too much if we have to process a huge set of data. Of course, one could use secondary memory, but this could significantly affect performance. Another workaround could be to perform several scans, for example one for each collector peer, but we have to consider that each scan can take up to a dozen minutes.

To tackle the above space complexity problem, we devised Algorithm 1. Given an update stream, it can pinpoint alledged table transfers from any collector peer, with a reasonable approximation of the start time of the transfer and its duration.

We distinguish between *pumping* and *vacuum* table transfers. The first corresponds to the typical large set of announcements from a router that notifies a neighbor about the full routing table. The latter corresponds to an explicit withdrawal of a large set of prefixes, potentially the full routing table. This should never happen, according to the RFC specifications; in fact, there is no state transition in the BGP finite state machine that can cause the action of withdrawing all the known prefixes; therefore, while definitely being table transfers, this kind of events are symptomatic of something unusual happening between the route collector and the collector peer. On the other hand, experience shows that this kind of phenomena can happen with non negligible frequency.

Algorithm 1 is composed by four steps. In Step 1, we define a function ρ that provides the number of prefixes to check against to decide if a table transfer occurred; the algorithm is parametric with respect to that function. Step 2 is used to compute the collections P_{RA} and P_W of the recently re-announced and recently widthdrawn prefixes, respectively. Steps 3 and 4 analyze the above sets, compare their cardinalities with ρ, and take decision about the presence of table transfers. We actually use the weighted average of the RIB size of the collector peer at time t_k as our ρ function, the weights being the amounts of time within which the RIB had a certain size. This choice has several advantages: it is computable without memory penalties (only one value per collector peer needs to be retained in memory), and it is hijacking-insensitive (i.e., even large prefix hijackings have a low influence on ρ). Namely, let t_i be the time of the i-th change of the RIB size of cp, and let $R(t_i)$ be the size of that RIB at time t_i, then

$$\rho(cp,t) = \frac{\sum_{i=1}^{k} R(t_{i-1}) \cdot (t_i - t_{i-1})}{t_k - t_0}.$$

Using Algorithm 1, we are able to identify occurring table transfers in a single sweep of all the updates ($O(n\omega\sigma)$ time), and only using $O(\omega\sigma)$ memory space. This in practice enables us to account for table transfers of all the considered collector peers at the same time. Our experiments show that we are able to process the data in the reference week in less than half the time spent with [28]. On the other hand the algorithm in [28] can be in some cases more accurate in identifying the exact beginning of the table transfer.

Table 5 summarizes the most relevant performance information of our implementation of Algorithm 1.

Algorithm 1 Identify Table Transfers

```
1. Define a weighting function $\rho(cp, t)$,
   $\rho : (\mathit{CollectorPeers} \times \mathit{Times}) \rightarrow \mathbb{R}$ that provides a numerical value to check against, based on the RIB size of peer $cp$ at time $t$.
2. For each update $u$
   • Compute the set $P_{RA}$ of prefixes such that $p \in P_{RA} \iff \ell_{u.cp}(p, u.t).t \geq u.t - \omega_r$, $\ell_{u.cp}(p, \ell_{u.cp}(p, u.t).t).\pi \neq \phi$ and $\ell_{u.cp}(p, u.t).\pi \neq \phi$. $P_{RA}$ is the set of prefixes which were already known by $u.cp$, and were reannounced, either with the same path or a different one, within a window of $\omega_r$ seconds before $u.t$.
   • Compute the set $P_W$ of prefixes such that $p \in P_W \iff \ell_{u.cp}(p, u.t).t \geq u.t - \omega_w$, $\ell_{u.cp}(p, \ell_{u.cp}(p, u.t).t).\pi \neq \phi$ and $\ell_{u.cp}(p, u.t).\pi = \phi$. $P_W$ is the set of prefixes which were already known by $u.cp$, and were withdrawn within a window of $\omega_w$ seconds before $u.t$.
3. For each time such that $| P_{RA} | \approx \rho(u.cp, u.t)$, we say there was a pumping table transfer for $u.cp$ which started approximately at $\min_{p \in P_{RA}} \ell_{u.cp}(p, u.t).t$, and lasted at least till $u.t$.
4. Similarly, for each time such that $| P_W | \approx \rho(u.cp, u.t)$, we say there was a vacuum table transfer for $u.cp$ which started approximately at $\min_{p \in P_W} \ell_{u.cp}(p, u.t).t$, and lasted at least till $u.t$.
```

Execution (wall clock) time	11:30:32 (hh:mm:ss)
Memory used (peak)	1.2 Gbyte

Table 5: Algorithm 1 - Time and memory performance

6.2 Computation of Local and Global Ranks

As we have seen in Section 5, being able to compute local and global ranks of each BGP peering is a crucial requirement for tracking the soundness of a peering over time, and spot where potential problems with the peering may have happened.

We defined Algorithms 2 and 3 to compute local and global ranks of each edge.

The algorithms maintain a structured representation of the running RIB of each collector peer, processing one update at a time. Whenever a change in the RIB is detected, local ranks of all edges contained in the new AS-path (if any) are increased, while local ranks of all edges contained in the old AS-path (if any) are decreased. Algorithm 3 is based on the computation of function $\lambda(edge, u.p, u.t)$. Such a function counts the number of unique collector peers whose route to prefix $u.p$ includes $edge$ at time $u.t$. The global rank of any edge is then increased or decreased if this counter changes, respectively, from 0 to non-zero or viceversa.

Algorithm 2 Compute Local Ranks

```
1. For each update $u$, considered for increasing values of $u.t$, consider its previous aspath $\ell_{u.cp}(u.p, u.t).\pi$
   • if $\ell_{u.cp}(u.p, u.t).\pi \neq \phi$ then $\forall edge \in \ell_{u.cp}(u.p, u.t).\pi$ decrease $lrank(edge, u.cp, u.t)$ by 1.
   • if $u.\pi \neq \phi$ then $\forall edge \in u.\pi$ increase $lrank(edge, u.cp, u.t)$ by 1.
```

While Algorithm 2 is quite simple, its implementation should be careful, to ensure sub-linear memory complexity and linear time complexity. The former can be accomplished noting that, for any cp, we can just consider the last change of a given prefix and the last value of the $lrank$ of an edge, thus only that piece of information should be kept in memory. The latter requirement can be fulfilled by writing each change in the $lrank$ to an output file before overwriting it with the new value; this assures that the algorithm never needs to consider the same update twice.

Since Algorithm 3 shares relevant information with Algorithm 2, it is efficient to run them together within the same program, as long as one can keep all the needed information in memory. Our experience showed this is still possible even considering all the collector peers from the RIS and ORV projects; should new projects or peers be

Algorithm 3 Compute Global Ranks

```
1. For each update $u$, considered for increasing values of $u.t$, consider its previous aspath $\ell_{u.cp}(u.p, u.t).\pi$
    • if $\ell_{u.cp}(u.p, u.t).\pi \neq \phi$ then
        – $\forall edge \in \ell_{u.cp}(u.p, u.t).\pi$, consider the number $\lambda(edge, u.p, u.t)$ of unique collector peers having
          an aspath including $edge$ in their current RIB for prefix $u.p$ and decrease it by 1 (in fact, the current
          collector peer does not use $edge$ any more to reach $u.p$)
    • if $u.\pi \neq \phi$ then
        – $\forall edge \in u.\pi$, increase $\lambda(edge, u.p, u.t)$ by 1 (in fact, the current collector peer now uses $edge$ to
          reach $u.p$)
2. For each time $u.t$ such that the value $\lambda(edge, u.p, u.t)$ changes,
    • if it changes from 0 to 1, then increase $grank(edge, u.t)$ by 1 (in fact, the current peer now uses an edge
      that no one used before for prefix $u.p$)
    • if it changes from 1 to 0, then decrease $grank(edge, u.t)$ by 1 (in fact, that edge is no more used to
      reach prefix $u.p$)
```

added, maybe splitting the rank-computing algorithms into separate programs would be needed.

Table 6 reports the performance of these Algorithms, referring to the single-program implementation. It is important to note that our Algorithms can be implemented with reasonable time and memory requirements, allowing them to run on virtually any machine and needing no dedicated hardware.

Execution (wall clock) time	13:48:43 (hh:mm:ss)
Memory used (peak)	2.7 Gbyte

Table 6: Algorithms 2 and 3 - Time and memory performance

6.3 Computation of Flow Changes

Local rank variations on a single path can be further explored examining how routes changed in a certain time interval.

We defined Algorithm 4, that, given a path π, a collector peer cp and a time window $[t_a, t_b]$, computes how flows of prefixes changed from/to π with respect to the given cp, exploiting the properties mentioned in Section 5. Formally, the algorithm computes $\Delta_{t_a}^{t_b} \hat{f}_P(cp, (u, w))$, as defined in 5.3.

Algorithm 4 Compute flows

```
1. Consider the collection $C$ of all the changes affecting prefixes in $P$ during the interval $[t_a, t_b]$.
2. For each change $c \in C$, let $sub\_old$ be the longest left subsequence that $c.\pi_{old}$ shares with $\pi$
    • for each edge $e$ in $sub\_old$, decrease $\Delta_{t_a}^{t_b} \hat{f}_P(e)$ by 1
    • for each edge $e$ in $c.\pi_{new}$, increase $\Delta_{t_a}^{t_b} \hat{f}_P(e)$ by 1
3. For each change $c \in C$, let $sub\_new$ be the longest left subsequence that $c.\pi_{new}$ shares with $\pi$
    • for each edge $e$ in $sub\_new$, increase $\Delta_{t_a}^{t_b} \hat{f}_P(e)$ by 1
    • for each edge $e$ in $c.\pi_{old}$, decrease $\Delta_{t_a}^{t_b} \hat{f}_P(e)$ by 1
```

This algorithm obviously can't be run in a batch fashion, so the implementation must satisfy strict time requirements; Table 7 reports the average execution time, calculated over a randomly chosen sample collection, in which each sample has $t_a = t_b$. While these results show that it is still impossible to pre-compute flows for each and every update, the execution time of a single query is still quite reasonable. However, recall this is still an experimental implementation, and we believe that optimizing data structures and the query management system could result in a further speed gain.

Average Query Execution time	40 sec
Memory used (peak)	2.7 Gbyte

Table 7: Algorithm 4 - Time and memory performance

6.4 GUI

Our GUI is integrated with software imported from the graphical interface of BGPlay [8] (Figure 5). That interface allows to look at the sequence of the updates collected by the collector peers. Users can select a specific path change c and can ask more information about c. This activates our service.

Following the development of [31], that underlined the importance of studying route changes in the framework of the Internet hierarchy, the first purpose of the GUI is to put c in the correct customer-provider perspective. This is especially important since the event causing c is more or less interesting for the ISP of $c.p$ depending on the location of the event in the hierarchy.

Gao has shown in [10] that each AS-path is composed by three (possibly empty) subpaths: the first climbs up the hierarchy from customers to providers, the second one is composed by a single peer-to-peer relationship, and the third path descends the hierarchy from providers to customers. This holds for almost every AS-path, so it should hold even for $c.\pi_{old}$ and $c.\pi_{new}$. If the event causing c happened in the climbing part of the paths, or in the peer-to-peer part, it is much more interesting for the ISP, since it could influence its future choices in terms of peerings and commercial agreements. Otherwise, if the event is in the descending part of the paths, the possibility of the ISP to control that portion of the Internet is so small that the event becomes negligible.

Discovering customer-provider relationships has been proved to be a hard problem ([23, 20, 26]), but several heuristics have been proposed in the literature giving reasonable results. However, our service is completely independent on the way the hierarchy is computed, so we regard it as an input. The automatic layout facility draws the ASes in $c.\pi_{old}$ and in $c.\pi_{new}$ according to the hierarchy. The layout we propose places top-level providers on the top and customers aligned on lower levels.

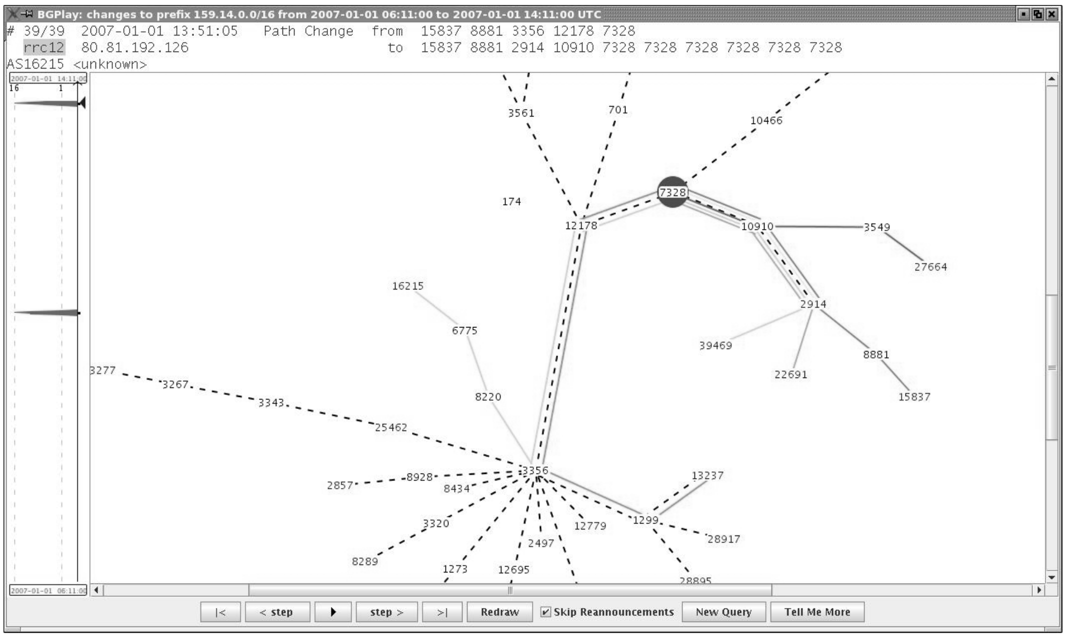
An example of usage of the GUI is shown in Figures 6-5. Figure 5 shows how BGPlay visualizes a path change. Clicking on the button “Tell me more”, the user enters our service. Figure 6 shows the same path change of Figure 5 within the customer-provider hierarchy. The left window lists the location of the relevant collector peers and gives information about their reliability. The right side buttons are used to deal with ranks and flows. As a side note, Figures 2 and 3 have been produced directly from the GUI.

7 Conclusions and Future Work

Despite the large amount of investigation, finding the causes of interdomain routing instabilities is still an elusive goal. On one hand the researchers and the network administrators have nowadays at disposal a very large set of BGP updates, recorded by several BGP collectors spread worldwide. On the other hand the methodologies for analyzing such data are not completely satisfactory. Further, there is no available public service for that.

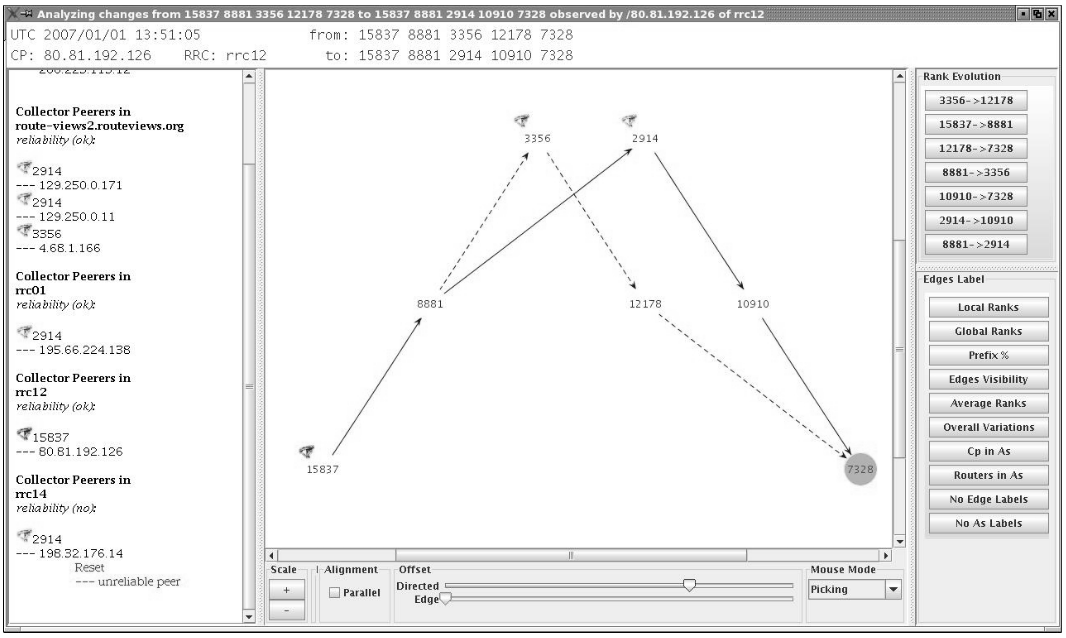
This paper gives three fundamental contributions to solve the above problem. First, we show a new simple model for describing BGP updates based on a flow network. Second, relying on the model, we present a methodology that tackles the root cause analysis problem from a new perspective. Namely, instead of looking at an entire chunk of updates searching major causes of instability we concentrate on a single BGP update. Our point of view is the one of an ISP that has seen an update on the route used by one of its prefixes and would like to have explanations only about it. Our methodology shows that this new perspective opens the opportunity, in many cases, to clarify which portion of Internet is responsible for the update. Third, we describe an on-line service that supports the methodology with many tools. In order to implement the service we had to overcome many computational problems. Hence, as a side effect of such an effort, we present in this paper algorithms that improve the performance of other problems already presented in the literature.

Obviously, we shall continue working on the methodology, enlarging the set of patterns we are able to recognize, in order to give full explanations of more updates. We shall also continue working on the service, implementing more tools to support the methodology. However, there are many further areas for future work. Namely, even if having a complete knowledge of the internal structure of the ASes from the outside is unfeasible, the studies on



```
BGPlay: changes to prefix 159.14.0.0/16 from 2007-01-01 06:11:00 to 2007-01-01 14:11:00 UTC
# 39/39  2007-01-01 13:51:05   Path Change  from  15837 8881 3356 12178 7328
  rrc12  80.81.192.126                      to    15837 8881 2914 10910 7328 7328 7328 7328 7328 7328
AS16215 <unknown>
```

Figure 5: A path change as observed through BGPlay.



```
Analyzing changes from 15837 8881 3356 12178 7328 to 15837 8881 2914 10910 7328 observed by /80.81.192.126 of rrc12
UTC 2007/01/01 13:51:05         from: 15837 8881 3356 12178 7328
CP: 80.81.192.126    RRC: rrc12   to: 15837 8881 2914 10910 7328
```

Collector Peerers in route-views2.routeviews.org
reliability (ok):

2914
--- 129.250.0.171
2914
--- 129.250.0.11
3356
--- 4.68.1.166

Collector Peerers in rrc01
reliability (ok):

2914
--- 195.66.224.138

Collector Peerers in rrc12
reliability (ok):

15837
--- 80.81.192.126

Collector Peerers in rrc14
reliability (no):

2914
--- 198.32.176.14
Reset
--- unreliable peer

Figure 6: A path change analyzed with our tool.

the route diversity (see, e.g. [16]) allow capturing, at least partially, the number of border routers an AS uses for EBGP peerings. Is it possible to combine our model and methodology with that information? Also, our flow model is valid for each single collector peer. Are there cases or assumptions when it is valid for collections of collector peers? An answer to this question could improve the analysis capability of the methodology.

References

[1] BGP Inspect. `http://bgpinspect.merit.edu/` .

[2] LinkRank. `http://linkrank.cs.ucla.edu/` .

[3] Oregon Route Views Project. `http://www.routeviews.org/` .

[4] Routing Information System. `http://www.ripe.net/ris/index.html` .

[5] Dion Blazakis, Manish Karir, and John S. Baras. BGP-Inspect - Extracting Information from Raw BGP Data. In *IEEE/IFIP NOMS*, 2006.

[6] Matthew Caesar, Lakshminarayanan Subramanian, and Randy H. Katz. Towards Localizing Root Causes of BGP Dynamics. Technical Report UCB/CSD-04-1302, 2003.

[7] Di-Fa Chang, Ramesh Govindan, and John Heidemann. The Temporal and Topological Characteristics of BGP Path Changes. In *ICNP*, 2003.

[8] Lorenzo Colitti, Giuseppe Di Battista, Federico Mariani, Maurizio Patrignani, and Maurizio Pizzonia. Visualizing Internet Routing with BGPlay. In *Journal of Graph Algorithms and Applications*, 2005.

[9] Anja Feldmann, Olaf Maennel, Z. Morley Mao, Arthur Berger, and Bruce Maggs. Locating Internet Routing Instabilities. *ACM SIGCOMM Computer Communication Review*, 2004.

[10] Lixin Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking*, 2001.

[11] Hongwei Kong. The Consistency Verification of Zebra BGP Data Collection. Technical report, 2003.

[12] C. Labovitz, R. Wattenhofer, S. Venkatachary, and A. Ahuja. The impact of Internet policy and topology on delayed routing convergence. In *IEEE INFOCOM*, 2001.

[13] Mohit Lad, Dan Massey, and Lixia Zhang. Link-Rank: A Graphical Tool for Capturing BGP Routing Dynamics. In *IEEE/IPIF NOMS*, 2004.

[14] Mohit Lad, Dan Massey, and Lixia Zhang. Visualizing Internet Routing Changes. *IEEE Transactions on Visualization and Computer Graphics*, 2006.

[15] Mohit Lad, Akash Nanavati, and Lixia Zhang Dan Massey. An Algorithmic Approach to Identifying Link Failures. In *PRDC*, 2004.

[16] Wolfgang Muhlbauer, Anja Feldmann, Olaf Maennel, Matthew Roughan, and Steve Uhlig. Building an AS-Topology Model that Captures Route Diversity. *ACM SIGCOMM Computer Communication Review*, 2006.

[17] Dan Pei, Matt Azuma, Dan Massey, and Lixia Zhang. BGP-RCN: improving BGP convergence through root cause notification. *Comput. Netw. ISDN Syst.*, 2005.

[18] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). IETF RFC 1771, 1995.

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

[20] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. In *IEEE INFOCOM*, 2002.

[21] Renata Teixeira and Jennifer Rexford. A Measurement Framework for Pin-Pointing Routing Changes. In *ACM SIGCOMM Workshop on Network Troubleshooting*, 2004.

[22] Soon Tee Teoh, Ke Zhang, Shih-Ming Tseng, Kwan-Liu Ma, and S. Felix Wu. Combining Visual and Automated Data Mining for Near-Real-Time Anomaly Detection and Analysis in BGP. In *VizSEC/DMSEC*, 2004.

[23] Feng Wang and Lixin Gao. On Inferring and Characterizing Internet Routing Policies. In *ACM SIGCOMM Conference on Internet Measurement*, 2003.

[24] Feng Wang, Zhuoqing Morley Mao, Jia Wang, Lixin Gao, and Randy Bush. A Measurement Study on the Impact of Routing Events on End-to-End Internet Path Performance. *ACM SIGCOMM Computer Communication Review*, 2006.

[25] Jian Wu, Zhuoqing Morley Mao, Jennifer Rexford, and Jia Wang. Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network. In *NSDI*, 2005.

[26] Xenofontas Dimitropoulos and Dmitri Krioukov and Marina Fomenkov and Bradley Huffaker and Young Hyun and kc claffy and George Riley. AS Relationships: Inference and Validation. *ACM SIGCOMM Computer Communication Review*, 2006.

[27] Kuai Xu, Jaideep Chandrashekar, and Zhi-Li Zhang. A First Step to Understand Inter Domain Routing Dynamics. In *ACM SIGCOMM MineNet Workshop*, 2005.

[28] Beichuan Zhang, Vamsi Kambhampati, Mohit Lad, Daniel Massey, and Lixia Zhang. Identifying BGP Routing Table Transfers. In *ACM SIGCOMM MineNet Workshop*, 2005.

[29] Jian Zhang, Jennifer Rexford, and Joan Feigenbaum. Learning-Based Anomaly Detection in BGP Updates. In *ACM SIGCOMM MineNet Workshop*, 2005.

[30] Ke Zhang, Amy Yen, Xiaoliang Zhao, S.Felix Wu, Dan Massey, and Lixia Zhang. On Detection of Anomalous Routing Dynamics in BGP. In *Networking*, 2004.

[31] Xiaoliang Zhao, Beichuan Zhang, Daniel Massey, Andreas Terzis, and Lixia Zhang. The Impacts of Link Failure Location on Routing Dynamics: A Formal Analysis. In *ACM SIGCOMM Asia Workshop*, 2005.