Computer Networks 110 (2016) 31–45

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Measurement of large-scale BGP events: Definition, detection, and analysis

Meng Chen[a,*], Mingwei Xu[a], Qing Li[b], Yuan Yang[a]

[a] *Dept. of Computer Science and Technology, Tsinghua University, China*
[b] *Graduate School at Shenzhen, Tsinghua University, China*

ARTICLE INFO

Article history:
Received 18 May 2016
Revised 18 August 2016
Accepted 20 September 2016
Available online 21 September 2016

Keywords:
BGP
Measurement
Anomaly detection

ABSTRACT

Measurement on the Border Gateway Protocol (BGP) system is important for understanding the Internet. Many attempts have been made to detect anomalous Internet events through dissecting BGP updates and tables. We notice that most works in this field either deploy/use few monitors or analyze aggregated statistics. Such practices may result in overestimating the impact of monitor-local events, which can be viewed by only a small area.

We propose Large-scale BGP Event (LBE), which affects many IP prefixes (high impact) and is widely observable (non-local). To detect LBE, we propose the Update Visibility Matrix (UVM) to record the prefix and monitor related to each update. We formulate the problem of identifying LBE in UVM, which is NP-hard. Then we propose a heuristic algorithm to solve it. We apply the scheme to 2.18 TB of BGP updates and find that the identified LBEs are highly correlated with many well-known disruptive incidents. Besides, we identify 101 LBEs that have never been investigated before. By conducting case studies, we find that the LBEs have high impact and are caused by various reasons. Our work can assist in network/Internet management tasks such as problem prevention, diagnosis, and recovery.

1. Introduction

Border Gateway Protocol (BGP) is the de facto inter-domain routing protocol. It is a path vector protocol, enabling the exchange of routing and reachability information among tens of thousands of Autonomous Systems (ASs, one or more networks under the control of a single administrative entity) on the Internet. The stability and robustness of the BGP system is always an important topic. With decades of effort, the BGP system is robust under most circumstances. However, big disruptive events can still seriously affect the connectivity and performance of the system. Therefore, a wealth of measurement works on detecting anomalous BGP dynamics have been proposed, e.g., [1–4]. Fast detection of disruptive events helps network operators to make timely and effective reactions, such as reconfiguring routing policies and enabling backup links; it also helps diagnosing and debugging network problems and helps network protocol enhancement and improvement.

We find that most of the works on BGP measurement leverage aggregated data from BGP monitors (i.e., BGP routers that provide route information), instead of investigating data from each monitor separately. Examples of aggregated data are total update quantity [5], overall update patterns [1], and the feature traces extracted from all updates [4]. Using aggregated data save much time in data-processing. However, it may lead to misunderstanding BGP instabilities, e.g., over-evaluating the impact of monitor-local events, which are visible to only a rather limited area of the Internet. According to Feldmann et al., 'BGP indeed provides significant isolation against routing updates' [6].

Primarily due to the incremental characteristic of BGP, local events are prevalent in the Internet. Specifically, after the initial exchange of the complete routing information, a pair of BGP routers exchange only the changes to that information. This characteristic largely restricts the propagation scale of updating information: it does not propagate far unless it changes the best routes at most of the BGP routers it traverses. In addition, route export policy also restrains the propagation scale of BGP updates. For example, a transit Internet Service Provider (ISP) usually does not export routes learnt from peers and providers to other peers and providers, hence it does not send the updates for these routes in these directions.

On the other hand, an event being widely observable does not necessarily mean it has high impact. For instance, individual new IP prefix announcement and individual IP prefix withdrawal by their origin ASs do not impair the BGP system. Such events affect

* Corresponding author.
E-mail address: chenm11@mails.tsinghua.edu.cn (M. Chen).

http://dx.doi.org/10.1016/j.comnet.2016.09.018
1389-1286/© 2016 Elsevier B.V. All rights reserved.

a small number of IP prefixes. Although some of these events can be rather disruptive, such as the hijacking of an important prefix, most of the time the impact of them is supposed to be trivial.

This paper focuses on detecting large-scale high-impact BGP event, i.e., the event that a) impacts a large quantity of prefixes, and b) can be observed by a large portion of (many) route monitors, within a given time period. We call such an event *Large-scale BGP Event (LBE)*. Essentially, an LBE indicates extraordinarily active changes in either the origin ASs of these prefixes or in a critical transit AS through which these prefixes are reachable.

LBE is anomalous and disruptive. Firstly, the large number of prefixes being updated is far beyond normal network operations. Secondly, the widely-observable characteristic means Internet-level inter-domain instability; in other words, the impact of LBE is non-local. Thirdly, the large numbers of monitors and prefixes imply an extremely high quantity of updates being propagated in the Internet, which not only deteriorate performance and connectivity on a large scale, but also is a potential threat to the processing capability of the routing facilities in the Internet.

In order to depict and detect LBE, we raise the concept of *Update Visibility Matrix (UVM)*, a binary matrix in which data from each monitor and for each prefix are recorded separately. We formulate the problem of identifying LBE in an UVM; basically, our target is to find a submatrix in the UVM so that it is dense and large. We prove the problem is NP-hard, and propose a heuristic algorithm, i.e., the Greedy Deletion Addition (GDA) algorithm, to solve it.

After setting the parameters, we apply the method to 2.18 TB data, including: a) the updates around twelve famous disruptive incidents, and b) the updates within ten months in 2013. The total BGP monitors exceed 400. After LBE identification, we analyze the features of the LBEs and conduct case studies to unveil the impact and cause of the LBEs.

Our major observations are concluded as follows.

1. There is a significant correlation between the well-known incidents and the identified LBEs, which highlights the effectiveness of our basic idea and method.
2. LBEs do exist even during ‘innocent period’. LBEs are not frequent and are quite unevenly distributed.
3. An LBE usually captures the majority of the updates within the time slot. Besides, The (prefix, monitor) pairs within an LBE show higher instability than those outside the LBE. Moreover, we find that our method is able to capture the complete procedure of most of the underlying incidents of the LBEs.
4. A single underlying incident could cause multiple LBEs (as many as 18) that are widely scattered in time (as long as two months).
5. In the case studies, we find that the impact of the underlying incidents is significant, and the major cause of them could be of various types, e.g, large scale rerouting, configuration error, and route leakage (or path spoofing).

The rest of the paper is organized as follows. Section 2 describes the related works. Section 3 elaborates on our method. Section 4 is dedicated to the measurement setup, including data collection, pre-processing, and parameter settings. In Section 5, we illustrate the identified LBEs for the two data sets. In Section 6, we conduct case studies on 23 LBEs in the 2013 data set. Finally in Section 7, we conclude the paper.

2. Related work

Measuring anomalous BGP events through analyzing BGP dynamics has remained an active area of research. In this section, we discuss the most relevant publications in this domain.

The closest work to ours is conducted by Comarela et al., who constructed a binary tensor to represent next-hop changes [3]. By identifying dense ‘cubes’ in the tensor, they detected large-scale coordinated rerouting events, which involve many ASs. Their data source is daily-collected routing tables, confining the analysis granularity to be daily. In comparison, we analyze BGP updates, which provide finer granularity so that we can detect abrupt short-term events. Besides, while they identified events that affect only tens of prefixes, we focus on events that affect at least thousands of prefixes. Thirdly, while their method focuses on only re-routing events, we detect various types of disruptive events.

Time interval methods. One of the pioneering works on detecting BGP events is conducted by Rexford et al. [7], who assumed that if two successive updates from the same peer for the same prefix are within an interval of 45 seconds, they belong to one event. Similarly, Wu et al. defined a BGP event as a sequence of BGP updates for the same prefix from a border router where the inter-arrival time is less than 70 seconds [8]. Sapegin and Uhlig proposed a similar approach to correlate BGP updates spikes into events [9]; specifically, if two spikes are close in time and have a large common prefix set, they are considered to be correlated. Compared with their methods, we define BGP events from the perspective of update propagation, and we focus on monitor participation and affected prefixes; our method can explicitly omit events with local impact.

Update pattern methods. Labovitz et al. raised a set of update patterns to measure the healthiness and stableness of the BGP system [10]. These patterns are essentially based on the content of successive BGP updates. For example, duplicate announcements for the same prefix indicate pathological behavior. Li et al. enhanced the pattern categorization method [11] and investigated the evolvement of the BGP system; they found that the BGP system had become much stabler. Moreover, Li and Brooks used the new set of pattern to identify the impact of famous Internet disruptive events [1]. In their work, patterns are organized into a vector to present the status of the Internet, and the deviation of a vector from a ‘normal range’ indicates anomaly. However, each element in the vector is the total sum of a pattern from all monitors, hence monitor-local events could potentially affect the result. In another work, Zhang et al. utilized a set of fixed patterns to search and identify routing anomalies [12]. They adopted five metrics to learn expected behaviors; the metrics are BGP update arrival frequency, number of AS paths in a period, updates type, AS path occurrence frequency, and AS path difference. Although per-monitor data are analyzed, the number of used monitors is lower than 10, providing a rather limited view of the Internet.

Statistics and machine-learning methods. Mai et al. raised a two-phase anomaly detection method [5], which basically applies wavelet analysis on a simple count of BGP updates. These updates are separated by the origin AS of IP prefix instead of by monitor. In this way, BGP withdrawals (no AS-path information) are excluded from analysis. Menon and Pottenger proposed a general framework, i.e., a higher order collective classifier, to detect and classify network events [13]. However, like many other works in this field, the impact of monitor-local events is not eliminated. Oliveira et al. adopted a simple count of total update to determine Highly Active Prefix [14], which is an important indicator of BGP anomalies. In their work, detailed distribution of the updates (which could be highly unbalanced) for the HAPs is not considered. Teoh et al. integrated visual and automated data mining to discover and investigate anomalies in Internet routing [15].

The scheme proposed by Deshpande et al. is based on adaptive segmentation of feature traces extracted from BGP update messages; it exploits the temporal and spatial correlations in the traces for robust detection of the instability events [4]. However, only four BGP monitors are used; so potential artifact (i.e., misinterpretation of measurement results) due to monitor-local dynamics could be strong. In contrast, we use more than 400 BGP monitors (among which >100 are suitable), and we believe these monitors are capable of detecting Internet-scale events. Liang and Zhang proposed a statistical technique based on pattern-matching to identify and locate familiar BGP routing instabilities [16]. Only 10 monitors were used, and data from these monitors were aggregated before applying the technique. We speculate there are three reasons for not using more monitors. First, to ease the effort needed to download, process, and store the data. Second, certain data source itself is small in scale. For example, BGPmon [17] provides real-time data, but the number of monitor is limited [18]. Third, the number of available monitors is small in early days [10].

Detection of individual event types. Mahajan et al. speculated BGP misconfiguration by observing short-time policy changes [19]. They varified their method by directly sending emails to ISP operators. Lad et al. analyzed the changes in the number and characteristics of BGP updates during the Slammer Worm attack [20], and discussed the impact of this incident. Dainotti et al. used multiple sources of data (both on the data-plane and control-plane) to study how the political turmoils in North Africa impair Internet connectivity [21]. Liu et al. proposed the metric 'betweenness centrality' to study the rerouting procedure after the 2011 Japan Tsunami [22]. Zou et al. focused on early detection of worms by developing a 'trend detection' model that fits the scenario well [23]. Siddiqui et al. leveraged both control-plane and data-plane observations to detect route leaks [24]. They established a theoretical framework to model route leakage and proposed three methods to detect it. By machine-learning the path change patterns, the scheme proposed by Ganiz et al. is able to classify two types of disruptive events, i.e., worm and blackout [25]. Unlike these works, we raise a unified method capable of identifying a wide scope of disruptions.

Summary. In terms of approach, in spite of a common 'theme', namely, the desire to detect high impact routing events, this paper differs from most previous works on multiple accounts. The major difference is the separation of the data from each monitor and for each prefix. In this way, we can effectively detect Internet-scale incidents. Secondly, we use data from a large number of monitors so that the results are not affected by the events local to a small area. Thirdly, our method has fine granularity and does not require the content of updates. Finally, our method is systematic and does not target a specific event type.

In terms of measurement results, a large portion of the previous works focus on testing their proposed detection methods on known incidents, and no new event is detected, e.g., [1,4,13,16]. Some works find new events that conform to specific criteria, e.g. [5,12]; but the nature and cause of the new events are not fully unveiled. Besides, some works both find new events and conduct detailed analysis on these events, e.g., [3,8]. Compared with most previous works, the measurement results of this paper is more comprehensive: in addition to testing our method on known events, we also detect new events and conduct in-depth analysis on them.

Simulation is important for studying giant and complex systems such as the BGP system. To integrate our method with BGP simulation tools (e.g., [26]), one needs to set up a special BGP router that peers with other BGP routers for the purpose of gathering updates. The major advantage of working with simulation tools, according to our understanding, is that each BGP router in a simulation tool could work as a monitor.

This paper is an extension of our previous works [27,28]. The major improvement in this paper is three-fold. First, we provide an in-depth analysis on the detected LBEs. We propose methods to infer the cause and analyze the impact of the LBEs. The large amount of new results significantly advance our understanding of LBE. Second, we also provide more materials and details, and some of the major new materials as follows: a) we add a comprehensive analysis on the heuristic algorithm; b) we investigate more famous incidents; c) we analyze the time pattern of the identified LBEs; d) we provide more details of the collectors; e) we provide an analysis of the clustering method to the parameter setting; f) we add an architecture figure to illustrate the whole measurement system. Third, this paper is of overall higher quality; we re-plot some figures and re-write much text content.

3. UVM-based LBE identification

In this section, we define UVM in Section 3.1, formulate the LBE identification problem in Section 3.2, and describe our heuristic algorithm in Section 3.3. We denote scalars by lower-case letters (a), sets by upper-case letters (I), vectors by lower-case bold-face letters ($\mathbf{v}$), and matrices by upper-case bold-face letters ($\mathbf{X}$).

3.1. Update visibility matrix

Definition 1. Within a given time period, an **Update Visibility Matrix (UVM)** is a binary matrix $\mathbf{X}$; its rows represent prefixes, and its columns represent monitors. Let x_{ij} be an element of $\mathbf{X}$; $x_{ij}=1$ if monitor j observes any BGP update for prefix i; otherwise $x_{ij}=0$.

Let P be the total prefix set and M be the total monitor set, and let $I \subseteq P$ and $J \subseteq M$ be subsets of prefixes and monitors, the pair (I, J) specifies a submatrix $\mathbf{X}_{IJ}$ in $\mathbf{X}$. Next, we define the attributes of $\mathbf{X}_{IJ}$.

Definition 2. Given an UVM $\mathbf{X}$, and a submatrix $\mathbf{X}_{IJ}$ in it, we define the following attributes.

The size of $\mathbf{X}_{IJ}$ is

$$Size(\mathbf{X}_{IJ}) = |I| \times |J| \tag{1}$$

The height and width of $\mathbf{X}_{IJ}$ are

$$Hei(\mathbf{X}_{IJ}) = |I| \tag{2}$$

$$Wid(\mathbf{X}_{IJ}) = |J| \tag{3}$$

The weight of $\mathbf{X}_{IJ}$ is the quantity of element '1':

$$Wei(\mathbf{X}_{IJ}) = \sum_{i,j} x_{ij} \tag{4}$$

The density of $\mathbf{X}_{IJ}$ is the proportion of element '1':

$$Den(\mathbf{X}_{IJ}) = \frac{Wei(\mathbf{X}_{IJ})}{Size(\mathbf{X}_{IJ})} \tag{5}$$

For simplicity, we assume that an UVM does not contain any empty (i.e., all '0') row or column. Note that the rows (and columns) of an UVM are not sequenced hence are interchangeable.

3.2. Problem formulation

Definition 3 ((θ_s, θ_w, θ_h, θ_d)**-Event).** Given an UVM $\mathbf{X}$, three non-negative integers θ_s, θ_w, θ_h and a real θ_d ($0 \le \theta_d \le 1$), A (θ_s, θ_w, θ_h, θ_d)-Event is a submatrix $\mathbf{X}_{IJ}$ in $\mathbf{X}$ such that $Size(\mathbf{X}_{IJ}) \ge \theta_s$, $Wid(\mathbf{X}_{IJ}) \ge \theta_w$, $Hei(\mathbf{X}_{IJ}) \ge \theta_h$, and $Den(\mathbf{X}_{IJ}) \ge \theta_d$.

For brevity, we denote (θ_s, θ_w, θ_h, θ_d)-Event as θ-Event. Given proper values of the thresholds, i.e., θ_s, θ_w, θ_h, and θ_d, a θ-Event is considered as a Large-scale BGP Event (LBE). We discuss the decision of the thresholds' values in Section 4.2. For an LBE, the four thresholds put restrictions in four aspects: θ_s decides the minimum size of an LBE; θ_w and θ_h prevent the detection of local or low-impact events; θ_d determines how much noise could be tolerated.

Definition 4 (Large-scale BGP Event Identification Problem)**.** Given an UVM $\mathbf{X}$, three non-negative integers θ_s, θ_h, θ_w, and a real θ_d ($0 \le \theta_d \le 1$), identify a submatrix $\mathbf{X}_{IJ}$ in $\mathbf{X}$ such that:

i) $\mathbf{X}_{IJ}$ is a (θ_s, θ_w, θ_h, θ_d)-Event; and
ii) *size*($\mathbf{X}_{IJ}$) is maximized.

The maximized size is for capturing the impact of the incident as completely as possible. Note that we assume at most one LBE within an analysis period. This is based on the following considerations. a) LBEs are supposed to be rare in the Internet; b) We do not assume an one-to-one mapping from an identified LBE to only one underlying incident. Considering the scale and complexity of the BGP system, we assume an LBE to be the superposition of multiple underlying incidents, and there is a major incident that is responsible for the majority of the instabilities. c) The length of each time slot is relatively short in our experiment, and we can get fine-granularity measurement result. We discuss the problem of detecting multiple LBEs in Section 3.3.3.

Theorem 1. *The large-scale BGP event identification problem is NP-hard.*

Proof. First, we set the thresholds θ_s, θ_w, and θ_h to 0, and set the threshold θ_d to 1. Next, we expand the input UVM $\mathbf{X}$ into a square matrix by adding 0s (to fill in the expanded area). Now we get a more-specific problem of the Large-scale BGP Event Identification Problem. For the more-specific problem, $\theta_d = 1$ means an LBE contains only element '1', so the identified LBEs before and after the expansion of $\mathbf{X}$ will be identical. Because the Maximum Clique Problem (MCP) [29] requires a symmetric matrix as input and a square and all-'1' submatrix as output, the more-specific problem we constructed (which does not put constraints on the input matrix) is equivalent to a more-general problem of the MCP. The MCP is NP-complete, so the more-specific problem we constructed is NP-hard, therefore the Large-scale BGP Event Identification Problem is NP-hard. □

To solve the LBE identification problem, we devise the following heuristic algorithm.

3.3. The greedy deletion addition (GDA) algorithm

The basic idea of the GDA algorithm has been presented in our previous work [28]. In this paper, we re-write some parts of the pseudo-code for clearer logic. Besides, the description of the algorithm is more comprehensive, and at the end of this subsection we provide a discussion of the algorithm.

The idea of the algorithm is as follows. Given the thresholds and an UVM $\mathbf{X}$, we conduct multiple steps of row/column deletion (mostly) or addition to get a submatrix $\mathbf{X}_{IJ}$ with a density greater than θ_d. The deletion action is for removing the row or column with the lowest density in order to increase density. The addition action is to add a previously-deleted row or column if its density becomes larger than that of the current submatrix, or adding it does not violate θ_d. When the algorithm terminates, if $Size(\mathbf{X}_{IJ}) \ge \theta_s$, we record $\mathbf{X}_{IJ}$ as an LBE; otherwise no LBE is detected. The pseudo-code of the GDA algorithm is presented in Algorithm 1. Algorithm 2 demonstrates the *Addition*() function, which is called in Algorithm 1.

3.3.1. Main iteration

The main iteration is from line 2 to 14. In each iteration, the action is either addition or deletion. We prefer addition to deletion because the optimization objective is maximized size. Whenever an addition does not decrease density, we conduct addition; otherwise we conduct deletion. The details of the addition and deletion actions are described below.

Algorithm 1 Greedy Deletion Addition Algorithm.

```
Input: UVM $\mathbf{X}$; thresholds $\theta_h$, $\theta_w$, $\theta_s$, and $\theta_d$
1:  $\mathbf{U} \leftarrow \mathbf{X}$, $den \leftarrow$ GET-DENSITY($\mathbf{U}$)
2:  while $den < \theta_d$ do
3:      $\mathbf{r}_a \leftarrow$ GET-CANDI-ADDROW($\mathbf{X}, \mathbf{U}$),$\mathbf{c}_a \leftarrow$ GET-CANDI-ADDCOL($\mathbf{X}, \mathbf{U}$)
4:      if not ADDITION($\mathbf{U}, \mathbf{r}_a, \mathbf{c}_a$) then
5:          $\mathbf{r}_d \leftarrow$ GET-CANDI-DELROW($\mathbf{U}, \theta_h$), $\mathbf{c}_d \leftarrow$ GET-CANDI-DELCOL($\mathbf{U}, \theta_w$)
6:          if $\mathbf{r}_d \ne NULL$ and $\mathbf{c}_d == NULL$ then
7:              $\mathbf{U}$.REMOVE($\mathbf{r}_d$)
8:          else if $\mathbf{c}_d \ne NULL$ and $\mathbf{r}_d == NULL$ then
9:              $\mathbf{U}$.REMOVE($\mathbf{c}_d$)
10:         else if $\mathbf{r}_d \ne NULL$ and $\mathbf{c}_d \ne NULL$ then
11:             $\mathbf{U}$.REMOVE(LARGER-INCDEN-PER-DELSIZE($\mathbf{r}_d$, $\mathbf{c}_d$))
12:         else
13:             Return $NULL$
14:     $den \leftarrow$ GET-DENSITY($\mathbf{U}$)
15:
16: $\mathbf{r}_a \leftarrow$ GET-CANDI-ADDROW($\mathbf{X}, \mathbf{U}$), $\mathbf{c}_a \leftarrow$ GET-CANDI-ADDCOL($\mathbf{X}, \mathbf{U}$)
17: while $\mathbf{r}_a \ne NULL$ or $\mathbf{c}_a \ne NULL$ do
18:     ADDITION($\mathbf{U}, \mathbf{r}_a, \mathbf{c}_a$)
19:     $\mathbf{r}_a \leftarrow$ GET-CANDI-ADDROW($\mathbf{X}, \mathbf{U}$), $\mathbf{c}_a \leftarrow$ GET-CANDI-ADDCOL($\mathbf{X}, \mathbf{U}$)
20:
21: $den \leftarrow$ GET-DENSITY($\mathbf{U}$)
22: while True do
23:     $\mathbf{r}_a \leftarrow$ GET-CANDI-ADDROW($\mathbf{X}, \mathbf{U}$), $\mathbf{c}_a \leftarrow$ GET-CANDI-ADDCOL($\mathbf{X}, \mathbf{U}$)
24:     $\mathbf{r}_a \leftarrow$ CHECK-FEASIBILITY($\mathbf{r}_a$, $\theta_d$), $\mathbf{c}_a \leftarrow$ CHECK-FEASIBILITY($\mathbf{c}_a$, $\theta_d$)
25:     if $\mathbf{r}_a == NULL$ and $\mathbf{c}_a == NULL$ then
26:         break
27:     $\mathbf{U}$.ADD(LARGER-INCSIZE-PER-DECDEN($\mathbf{r}_a$, $\mathbf{c}_a$))
28:
29: if $Size(\mathbf{U}) \ge \theta_s$ then
30:     Return $\mathbf{U}$
31: else
32:     Return $NULL$
```

Algorithm 2 Function: ADDITION($\mathbf{U}$, $\mathbf{r}_a$, $\mathbf{c}_a$).

```
Input: Submatrix $\mathbf{U}$; Candidate row $\mathbf{r}_a$ and column $\mathbf{c}_a$ for addition
1: if $\mathbf{r}_a \ne NULL$ and $\mathbf{c}_a == NULL$ then
2:     $\mathbf{U}$.ADD($\mathbf{r}_a$)
3: else if $\mathbf{c}_a \ne NULL$ and $\mathbf{r}_a == NULL$ then
4:     $\mathbf{U}$.ADD($\mathbf{c}_a$)
5: else if $\mathbf{r}_a \ne NULL$ and $\mathbf{c}_a \ne NULL$ then
6:     $\mathbf{U}$.ADD(LARGER-WEIGHT($\mathbf{r}_a$, $\mathbf{c}_a$))
7: else
8:     Return False
9: Return True
```

Addition. Among all the rows outside the submatrix $\mathbf{U} = \mathbf{X}_{IJ}$, i.e., $\mathbf{x}_{kJ}$ ($k \in K, K = P \setminus I$), we get the row with the maximum density, $\mathbf{x}_{\mu J}$. if $Den(\mathbf{x}_{\mu J}) \ge Den(\mathbf{X}_{IJ})$, $\mathbf{x}_{\mu J}$ is the candidate addition row $\mathbf{r}_a$. This job is accomplished by function GET-CANDI-ADDROW(). Similarly, function GET-CANDI-ADDCOL() gets the candidate addition column $\mathbf{c}_a$. These functions return *NULL* if no feasible candidate is found.

As shown in Algorithm 2, if both $\mathbf{r}_a$ and $\mathbf{c}_a$ are not *NULL*, we add to $\mathbf{X}_{IJ}$ whichever has greater weight. This is accomplished by function LARGER-WEIGHT() in the function ADDITION(). We adopt the metric weight (i.e., the quantity of '1') as the criteria because we have two concurrent goals: a) the size of $\mathbf{U}$ after addition is maximized, and b) the density of $\mathbf{U}$ after addition is maximized so that less deletion is conducted in future. Weight is a compromise of the two goals because it is the product of size and density.

Deletion. If an addition is conducted, the algorithm goes to the next iteration; otherwise the algorithm turns to deletion. We get the candidate deletion row and column (i.e., $\mathbf{r}_d$ and $\mathbf{c}_d$) by functions GET-CANDI-DELROW() and GET-CANDI-DELCOL() respectively. The scheme is as follows: among all the rows $\mathbf{x}_{iJ}$ ($i \in I$) in the submatrix $\mathbf{U} = \mathbf{X}_{IJ}$, $\mathbf{x}_{\gamma J}$ has the minimum density; if $Den(\mathbf{x}_{\gamma J}) < Den(\mathbf{X}_{IJ})$, $\mathbf{r}_d = \mathbf{x}_{\gamma J}$. Using a similar method, we get $\mathbf{c}_d$.

If both $\mathbf{r}_d$ and $\mathbf{c}_d$ are not *NULL*, we delete from $\mathbf{U}$ whichever maximizes the increased density per deleted size; this is achieved by function LARGER-INCDEN-PER-DELSIZE(). This criterion is due to a) the optimization objective being maximized size, and b) the need for higher density increase. Note that if the deletion violates θ_w or θ_h, the corresponding candidate is set to *NULL*.

If both addition and deletion are infeasible, the algorithm terminates (line 13).

3.3.2. Addition in the end

The main iteration ends when $Den(\mathbf{U}) \geq \theta_d$. Next, we conduct two more rounds of addition by different methods to maximize $Size(\mathbf{U})$. Round one starts and ends at line 16 and 19 respectively. In this round, if further addition does not reduce $Den(\mathbf{U})$ (note that the main iteration terminates as soon as $Den(\mathbf{U}) \geq \theta_d$), we apply the same addition scheme as that within the main iteration. Round two is from line 21 to 28. In this round, we allow $Den(\mathbf{U})$ to decrease, as long as $Den(\mathbf{U}) \geq \theta_d$. We check the feasibility of the candidates by CHECK-FEASIBILITY(): if adding a candidate violates θ_d, we set the candidate to *NULL*. After getting the candidates, we select the candidate that maximizes the increased size per decreased density, accomplished by function LARGER-INCSIZE-PER-DECDEN().

3.3.3. Discussion

Time complexity. The number of iterations in the main-iteration part is $O(|P| + |M|)$, given the prefix set and monitor set of $\mathbf{X}$ being P and M respectively. Note that addition in the main iterations is rare, orders of magnitudes less than deletion. This is due to the stringent condition for addition: while a row is deleted because it has the smallest density, it is added because its density becomes larger than $Den(\mathbf{U})$; but the value of $Den(\mathbf{U})$ keeps increasing.

Within an iteration, it takes $O(|M|)$ to get the candidate addition and deletion columns by simply scanning all the columns. By adopting a 'value to index mapping' data structure, it also takes $O(|M|)$ to get the row candidates, because the rows in and out of $\mathbf{U}$ have $O(|M|)$ different weight values. So the time complexity of the GDA algorithm is $O(|M|(|P| + |M|))$. Note that we assume the number of prefixes is (much) larger than that of monitor; otherwise one ought to replace the former M with P.

UVM pre-processing. In practice, the input UVM $\mathbf{X}$ could be rather sparse, and there could be a large quantity of rows and columns with rather few '1's, e.g., one or two '1's. Deleting those rows before running the GDA algorithm would significantly reduce the scale of the problem. Since we are mainly interested in the rows and columns with high density, the pre-processing does not impair the correctness of our method. Only in extreme situations, e.g., when the number of prefixes is small and the deleted elements could be components of an already-dense submatrix, may the pre-processing affect the outcome of the algorithm.

Half-way termination. The computation time could be further reduced by terminating the algorithm and returning *NULL* (i.e., no detection) as soon as $Size(\mathbf{U})$ becomes smaller than a predefined threshold, e.g., $0.8\theta_s$, in the main iterations. Since addition actions are rare compared with deletion actions, if $Size(\mathbf{U})$ is 'sufficiently small' it is unlikely that $Size(\mathbf{U})$ would increase to a value larger than θ_s on a later stage.

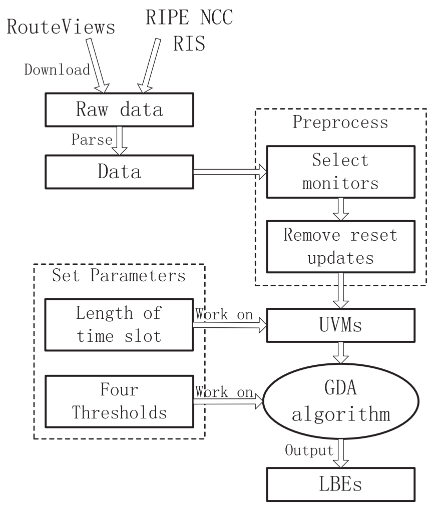


Fig. 1. The architecture of the measurement system.

Detecting multiple LBEs. If under special circumstances one needs to detect multiple LBEs within a single time period, running the GDA algorithm multiple times achieves this goal. Specifically, when each GDA run terminates, one should set the identified LBE $\mathbf{U}$ to '0's in the original UVM $\mathbf{X}$, and use the new UVM as the input and run the GDA algorithm again. The entire procedure terminates if no LBE is identified in a complete GDA job.

4. Measurement setup

The architecture of the measurement system is plotted in Fig. 1. In this section, we describe data collection, preprocessing, and parameter settings.

4.1. Data collection and preprocessing

We download BGP updates from RouteViews [30] and RIPE NCC [31] (Réseaux IP Européens is the French for "European IP Networks"; NCC stands for Network Coordination Centre). The RouteViews project and the Routing Information Service (RIS) project of RIPE NCC operate multiple route collectors. Each route collector has BGP sessions with multiple BGP-speaking routers (i.e., the BGP monitors) in the Internet. RIS publicizes BGP updates every 5 minutes, and RouteViews publicizes BGP updates every 15 minutes; both of them publicize routing tables every eight hours.

We use totally 17 collectors, which provide data from 452 monitors (on January 1st, 2013). The six RouteViews collectors are: route-views2, route-views4, route-views.eqix, route-views.isc, route-views.linx, and route-views.wide. The eleven RIS collectors are: rrc00, rrc01, rrc03, rrc04, rrc05, rrc10, rrc11, rrc12, rrc13, rrc14, and rrc15. The total update data add up to 2.18 TB.

4.1.1. Monitor winnowing

Some of the 452 monitors are unsuitable for our experiment thus are omitted in our analysis. a) **Partial view.** A monitor has a partial view means it can see only a limited number of IP prefixes. If a monitor has only a partial view of the Internet, it may miss some Internet-level BGP events. So we select only the monitors with global view. To achieve this goal, we set a benchmark as the quantity of prefixes in the routing table of a core router [32]. A global-view monitor should see at least 90% of the benchmark. This criterion removes 269 monitors. b) **Duplicate sessions.** Some monitors have sessions with more than one collector, hence are interpreted as multiple monitors, and this may lead to the impact observed by those routers being incorrectly amplified. To deal with the issue, we select only one session among the multiple ones with the same router. This step removes 52 monitors. c) **Unbalanced distribution.** Some ASs own multiple monitors, which may result in the impact of events local to these ASs being overstated. So we select only one monitor in each AS. This step further removes 8 monitors. In summary, among the 452 monitors, 123 are suitable for our measurement.

4.1.2. About the selected monitor ASs

Tier. It is a common practice to categorize ASs into tiers to indicate their importance to the interconnection of the Internet. We get the tier of the selected monitor ASs by the method proposed by Oliveira et al. [33], who adopted the size of customer cone as the metric. An AS's customer cone is the ASs that are reachable through AS-level provider-to-customer links. The size of customer cone indicates the influence of an AS. The method is: the ASs with less than 5 downstream customers are stubs; the ASs with between 5 and 50 downstream customers are small ISPs (tier-3); the remaining non-tier-1 ASs are large ISPs (tier-2); the list of tier-1 ISPs are obtained in the same way as [33]. We get customer cone data from a public repository [34], which is maintained by RIPE RIS. The result for the 123 monitors is, tier-1: 8, tier-2: 37, tier-3: 43, stub: 35.

Geographical location. Next, we map the ASs to countries/regions by using the geographic data from a website [35]. For a tier-1 AS, we record its region as 'Global'. We find that the 123 monitors are spread across 26 countries/regions in the world.

To summarize, the monitors are widely spread in the Internet. After setting a proper width threshold, our method is capable of detecting events that have Internet-scale impact.

4.1.3. Eliminating the effect of BGP session reset

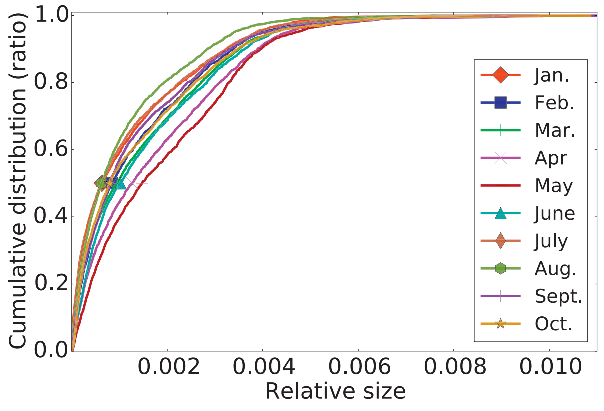
Sometimes the BGP session between a route collector and a route monitor is re-established. In this situation, the complete BGP table of the monitor is sent to the collector in the form of a large amount of updates. While these updates occupy notable computing and storage resource, they provide little information about global Internet status. In fact, BGP session reset is a special type of local event. Therefore, we delete the updates caused by BGP session reset through applying the method raised by Cheng et al. [36].

4.2. Parameter settings

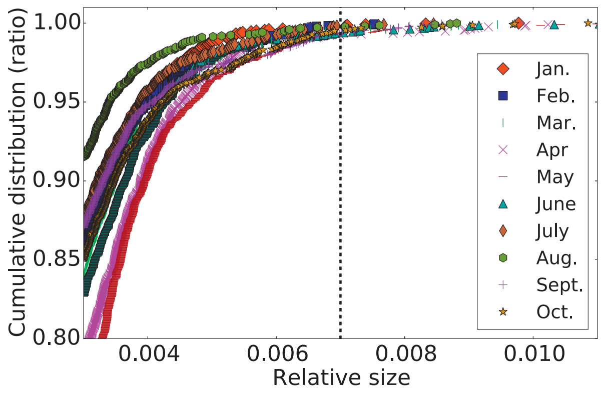
Because the quantities of working monitors and total prefixes are not constant, we set θ_w and θ_s according to the quantity of monitor and that of total prefix at the measurement time t, i.e., $|M_t|$ and $|P_t|$. Specifically, θ_w is a ratio of $|M_t|$, and θ_s is a ratio of $|M_t| \times |P_t|$. For brevity, in the following of this paper, we directly use the ratios to represent width and size, and call them *relative width* and *relative size* respectively.

4.2.1. Length of time slot

We divide a time period into a series of time slots to carry out the analysis. Based on the following considerations, we set the length of time slot to 20 minutes. First, we try to decrease the possibility that the impact of an underlying incident is divided into separate slots; this requires large slot length. Second, we try to decrease the number of concurrent underlying incidents within a single slot; this requires small slot length. While the BGP convergence delay depends on topology and BGP configurations [37,38], previous research has established that it usually takes several minutes or less time for a re-routing event to converge, and it is quite rare that the convergence time is longer than 10 minutes [39], or one hour [40]. Therefore, we believe 10, 20, and 30 minutes are among



(a) The complete CDFs.



(b) The zoomed part ($ratio > 0.8$).

Fig. 2. Cumulative distribution of θ-Event size. (We set $\theta_s = 0$ and run the GDA algorithm to get the complete size distribution of each month.)

the reasonable decisions. We use 20 minutes in this paper, and analyze the impact of varying this parameter in Section 5.4.

4.2.2. Density threshold

Our goal is to detect a 'dense' submatrix in an UVM, so we cannot set θ_d to a small value; otherwise the identified LBE becomes less interesting because of its sparsity. On the other hand, we cannot set θ_d to 1, because a full-'1' submatrix is unlikely to have a large size considering the noise and the non-full visibility of the prefixes and monitors in UVM. In our measurement, we set θ_d to 0.8. We show the effect of varying this threshold in Section 5.4.

4.2.3. Width threshold

In this paper, we set θ_w to 0.4 (i.e., $\theta_w = 40\% \times |M_t|$) considering that a) an LBE does not necessarily have to be observed by all of the monitors, and b) $\theta_w = 0.4$ effectively avoids that all of the monitors in an LBE are of the same tier or in the same country/region. For example, for the 123 monitors, the threshold is 49 monitors; according to the monitor-to-region categorization, at least 4 nations/regions observe an identified LBE. Besides, the monitors are from at least two tiers. We discuss the sensitivity of our method to θ_w in Section 5.2.1.

4.2.4. Size threshold

After setting the length of time slot, density, and width, next we determine θ_s. For each month within from January to October 2013, we plot in Fig. 2 the cumulative distribution of the size of the identified θ-Event when setting θ_s to zero. Fig. 2a shows the complete CDFs, and Fig. 2b shows the part where the ratio is larger than 0.8. As is evident in the figure, the results for the ten months are similar: the vast majority of the θ-Events have small size. We assume that the Internet is stable and healthy most of the time, so θ_s should be large enough so that all the analyzed months contain small proportions of LBEs. That is to say, we are interested in capturing the θ-Events in the tails of the curves, and θ_s should accommodate all the distribution curves. Fig. 2b illustrates that setting θ_s to 0.007 catches approximately the top 1% θ-Events in each month. Specifically, the vertical line at size 0.007 intersects the curves at ratios between 99.2% to 99.8%. Since this value suits our assumption well, we set θ_s to 0.007 in our experiment.

4.2.5. About height threshold

After setting θ_s, the height of an LBE is at least $\theta_s/Wid(\mathbf{X})$. Since in our experiment the quantity of monitor is orders of magnitudes smaller than that of prefix, $\theta_s/Wid(\mathbf{X})$ is supposed to be a large value (much larger than θ_w). Therefore, we omit the height threshold in our experiment. Note that if one uses a much larger set of monitors than ours, e.g., >1000, θ_h must be set to avoid the detection of low-impact event.

5. Measurement results

In this section, we apply our method to two sets of update data, including a) data around well-known disruptive incidents, e.g., blackout, and b) 10 months' data in 2013. We illustrate the identified LBEs associated with these incidents as well as the new LBEs in 2013; we analyze the patterns and characteristics of these new LBEs. Next, we adopt the DBSCAN technique to cluster the LBEs. After that, we demonstrate the impact of the major parameters. At the end of this section, we analyze the performance of our algorithm.

5.1. Result for the famous disruptive incidents

5.1.1. Overview

The identified LBEs for the disruptive incidents are demonstrated in Fig. 3. The blue markers indicate LBEs when $\theta_s = 0.007$, and the red markers indicate additional LBEs when decreasing θ_s to 0.006. The impact of most of these incidents on the Internet has been investigated in existent literature. To be Specific, seven of these incidents, including 3 a–g, are analyzed in [1]. The SEA-ME cable cut (3 i) is investigated by Chan et al. [41]. The Japan Tsunami is analyzed by Liu et al. [2]. We do not select the incidents that affect a small quantity of prefixes (e.g., individual prefix hijacking) or are known to have limited visibility (e.g., attack against a small content provider).

In the figures, x-axis denotes time, and y-axis denotes relative size. The vertical dash line in each figure marks the occurrence of the underlying incident. There are two special cases: a) The dash line in Fig. 3c denotes the date of the first landfall of the Hurricane. b) The accurate time of the SEA-ME cable cut is still unknown, and the dash line in Fig. 3i denotes the beginning of April 14th, 2010 (it is known that the cable cut happened on this day). Also note that while the occurrence time is a critical reference point, it is not necessarily the time at which the incident affects the BGP system. For example, the 2006 Taiwan earthquake did not immediately cut off undersea cables hence impair connection; the aftershocks did it.

5.1.2. The identified LBEs

As demonstrated in the figures, the quantity and size of the LBEs increase immediately/shortly after the occurrence of the incidents, indicating a high correlation between these LBEs and the disruptive incidents. The impact pattern of these incidents is different. In Fig. 3d, k and l, the impact is sudden and short-lived. In contrast, for most of the incidents the impact is persistent; the typical examples are shown in Fig. 3e–h, and j. In particular, we observe an impact escalation in Fig. 3e: the largest LBE occurs after more than 10 hours of the incident's occurrence.

Note that the results presented in Fig. 3a and b have different characteristics than the others. a) The number of LBEs is significantly higher than that in the other figures. b) A large portion of the LBEs exist before the occurrence of the incidents. c) The impact of the incidents is indicated by the sudden rise in the size of the LBEs. We believe the peculiar phenomena are due to the size threshold being too small for these incidents. Firstly, while the incidents cover a wide time span, the size threshold θ_s is determined by analyzing only the 2013 data set for the ease of data gathering and processing. Secondly, it is believed that the BGP system is more unstable in the early days [10,11]. Thirdly, a larger θ_s (e.g., 0.015) could both eliminate the LBEs before the dash lines and preserve the LBEs indicating the incidents.

Also note that if we decrease θ_s from 0.007 to 0.006, some LBEs exist before the dash lines in Fig. 3c, d and l. In such situations, it is apparent that the LBEs right after the dash lines are larger in size than those before the lines, especially in 3 d and l. Fig. 3c is different than the other two figures in that the largest LBE is far from the dash line. This is because we cannot decide an accurate impact time-point for a Hurricane event.

In summary, the measurement result of the famous incidents highlights the effectiveness of our method and basic idea. Next, we turn our focus to the performance of our method on an 'innocent' update set.

5.2. Result for the 2013 data set

The identified LBEs for the 2013 data set are shown in Fig. 4; the colored dots are clusters, which we describe later. We detect totally 101 LBEs within the period from January 1st to October 31st, 2013. That is averagely 2.33 LBEs per week. Therefore, generally speaking, LBEs are rare, which complies with the common belief that the BGP system is stable most of the time. However, the time distribution of these LBEs is quite uneven. For example, while there are more than 15 LBEs in both April and May, only 3 LBEs exist in February. We conduct a detailed analysis of 23 of these LBEs in Section 6.

Next, we take a closer look at the characteristics of these LBEs.

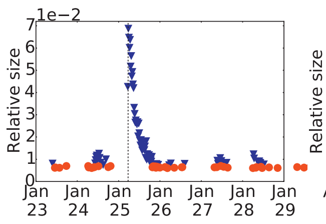
5.2.1. Width and size

The width of the 101 LBEs as a function of their size is plotted in Fig. 5. As is evident from the figure, most of the LBEs have rather large widths, which are significantly larger than the width threshold θ_w. Specifically, only 5 LBEs have width values smaller than 0.8, and only one LBE has width smaller than 0.5. The result implies that LBE identification is not sensitive to the width threshold.

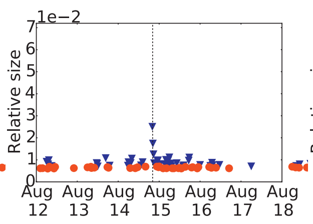
In terms of size, only 3 LBEs have size values larger than 0.013; their size values are 0.017, 0.02, and 0.032 respectively. On the other side, the size values of 85 LBEs are concentrated in the range [0.007, 0.01]. The result implies that the number of identified LBE is sensitive to the size threshold, which is also implied in Fig. 2b. Finally, the Pearson's correlation coefficient between the two variables is -0.13, which indicates no apparent correlation.

5.2.2. Update ratio

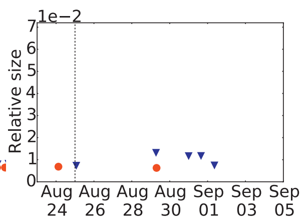
The ratio of updates contained in an LBE to all the updates observed in the corresponding time slot is a critical criteria to assess the reasonableness of the density threshold. If θ_d is too large, the identified LBE would be quite small, hence the updates in the LBE will also be small in quantity; in this way, our method cannot capture the major instabilities reflected in the UVM. However, if θ_d is



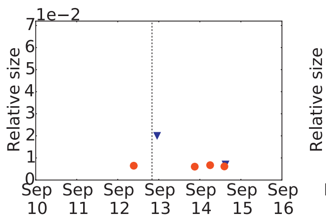
(a) 2003 Slammer worm.



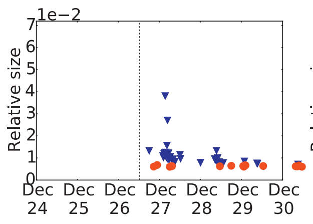
(b) 2003 US east-coast blackout.



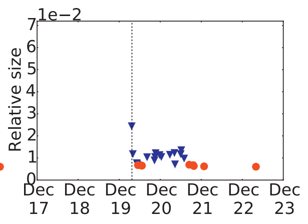
(c) 2005 Hurricane Katrina.



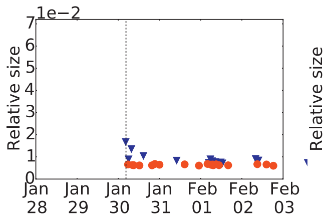
(d) 2005 LA blackout.



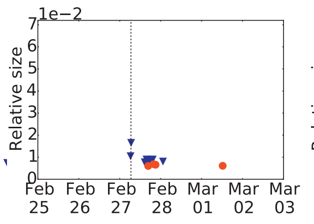
(e) 2006 Taiwan earthquake.



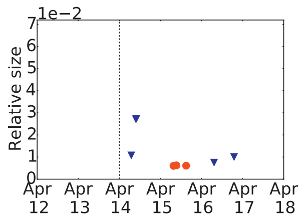
(f) 2008 Mediterranean cable cut I.



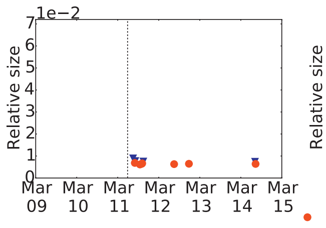
(g) 2008 Mediterranean cable cut II.



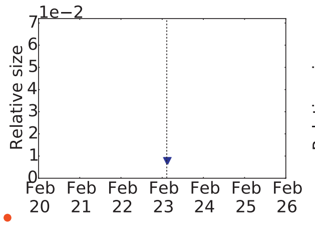
(h) 2010 Chile earthquake.



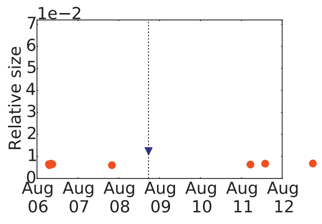
(i) 2010 SEA-ME cable cut.



(j) 2011 Japan Tsunami.



(k) 2012 Australia route leakage.



(l) 2012 Canada route leakage.

Fig. 3. Identified LBEs for the famous incidents. The dash lines mark the occurrence of these incidents. The blue markers indicate the LBEs when $\theta_s = 0.007$, and the red markers indicate the additional LBEs when decreasing θ_s to 0.006.

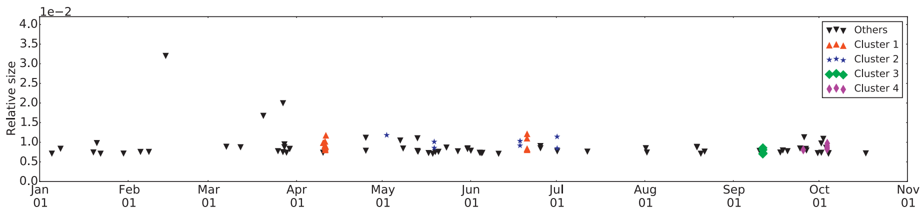


Fig. 4. Identified LBEs for the 2013 data set.

too small, the ratio could be very large, e.g., close to one. Furthermore, this ratio also shows the capability of an LBE in capturing instabilities.

Fig. 6 demonstrates this ratio as a function of the size of the LBE. It shows that all but one of the LBEs capture more than 50% of the updates in the corresponding time slot, and as many as 72.3% of the LBEs capture >70% of the updates. Besides, the ratios are not concentrated in a close-to-one range. The result indicates that setting θ_d to 0.8 is a reasonable decision. Finally, the Pearson's correlation coefficient between the two variables is 0.064, which indicates no apparent correlation.

5.2.3. *The number of updates in each '1' element*

In Fig. 7, we present the average number of updates in each '1' element inside and outside an LBE. A circle represents an LBE, the y axis is the average inside the LBE, and the x axis is the average outside the LBE. The figure shows the ratio of the two averages (in/out) can be as small as 0.314, and as large as 7.962. Although

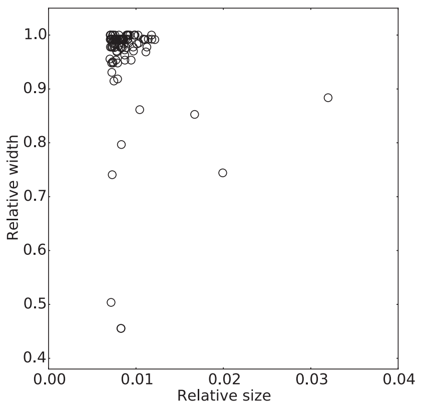


Fig. 5. Relative width as a function of relative size.

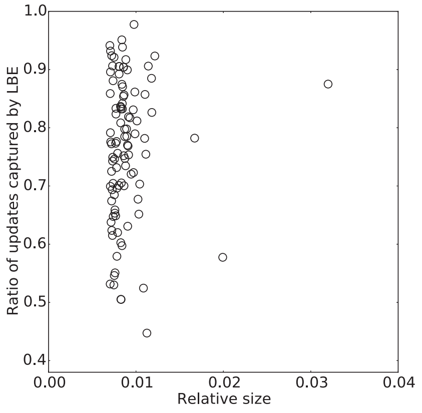


Fig. 6. The ratio of updates captured by the LBEs.

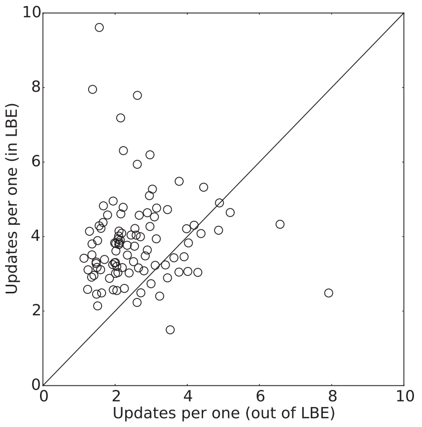


Fig. 7. The average number of updates captured by each ‘1’.

the range of the ratio is wide, most of the time slots have larger average in the ‘1’s in the LBEs; specifically, as shown in the figure, 83 LBEs are to the left of the line that crosses the figure. The result indicates that the unstable (prefix, monitor) pairs, which involve more updates than other pairs, tend to be captured by LBE.

5.2.4. Time pattern

For LBE $\mathbf{U}_t$ in time slot t, we denote its prefix set and monitor set as P_{ut} and M_{ut}. In the UVM $\mathbf{X}_{t-1}$ and $\mathbf{X}_{t+1}$, we get the density of the submatrix decided by P_{ut} and M_{ut} (if no corresponding row/column in the UVM, regard the elements in the row/column as ‘0’), denoted as $Den_{t-1}(P_{ut}, M_{ut})$ and $Den_{t+1}(P_{ut}, M_{ut})$. If $Den_{t-1}(P_{ut}, M_{ut}) \geq 0.4$ (0.4 is half of θ_d), we denote $\mathbf{U}_t$ as *p-persistent* (p stands for previous), and if $Den_{t+1}(P_{ut}, M_{ut}) \geq 0.4$, we denote $\mathbf{U}_t$ as *n-persistent* (n stands for next).

Since the persistency may be due to LBEs in successive time slots, we omit such cases in the analysis. In the result, we get 9 and 12 p-persistent and n-persistent LBEs respectively, and only 2 LBEs are in both groups. The result implies that for the majority of the LBEs, our method can capture their complete/major process.

In the case studies in Section 6, we apply the time pattern scheme to decide the time boundaries of the detailed analysis.

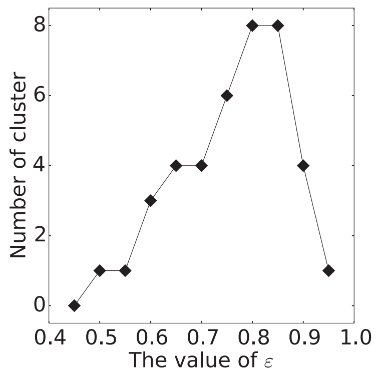


Fig. 8. The Number of cluster as a function of ε.

5.3. LBE clustering

In this part, we cluster the LBEs in the 2013 data set. The purpose of clustering is two-fold: 1) to illustrate whether there are multiple LBEs that are likely caused by the same reason; 2) to find the target LBEs for the case study. The clustering is based on the prefix set of the LBEs because a) for an LBE, its monitor set is orders of magnitudes smaller than its prefix set; b) the majority of the LBEs have large width, hence the difference is small; c) prefix set provides rich information about the cause of an LBE, which we describe in detail in the case studies.

We adopt the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) technique [42] in this part because 1) it does not require a priori number of clusters, and 2) it is simple (requires only two parameters) and quick. We admit that DBSCAN is not the only choice; other clustering techniques, e.g., CURE [43] and KNN [44], could also work in our scenario.

For two LBEs $\mathbf{A}_{I_1J_1}$ and $\mathbf{B}_{I_2J_2}$, we define the distance between them as the Jaccard distance between their prefix sets:

$$Dist(\mathbf{A}_{I_1J_1}, \mathbf{B}_{I_2J_2}) = 1 - \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|} \tag{6}$$

The distance between two LBEs with identical prefix sets is zero, and two LBEs with completely different prefix sets have a distance of one.

The DBSCAN technique requires two parameters, a) ε: the maximum distance to find directly reachable point, b) minPts: the minimum number of points to form a dense region. Fig. 8 shows the number of cluster as a function of the parameter ε. Note that one of our main purposes is to find large clusters for the case studies, so we set minPts to 4. According to the Figure, 0.45 seems to be too small, with none cluster found, and 0.8 seems to be too large. When $\varepsilon = 0.95$, most LBEs are grouped into only one cluster. In this part, we set ε to 0.65, which is a moderate value among these candidates. A distance value of 0.65 means that two LBEs have approximately half of their prefix sets in common.

As shown in Fig. 4, LBEs belonging to different clusters are marked with different colors; black marks unclustered LBEs. We find four clusters, containing 18, 7, 4, and 5 LBEs respectively. While some clusters persist for a short time, some span months.

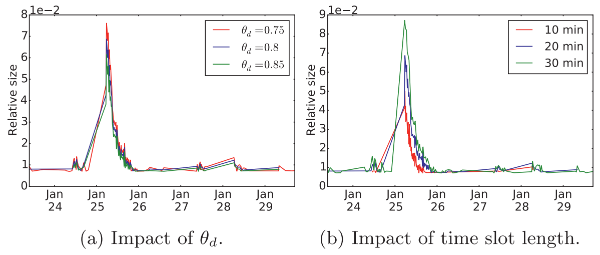


Fig. 9. Impact of parameters on the LBE detection for the Slammer worm incident.

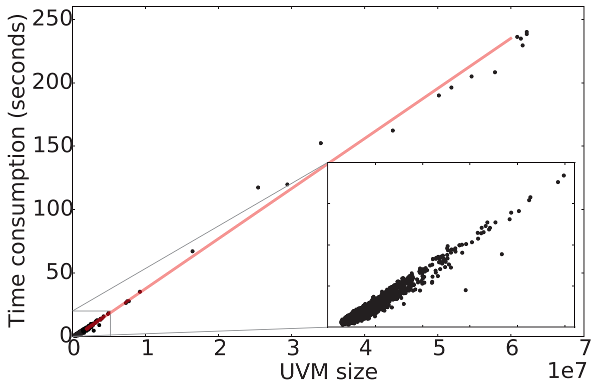


Fig. 10. Time consumption as a function of UVM size.

The result indicates that some Internet incidents could generate multiple LBEs, which may be widely scattered in time. To understand the reason for the clusters, we conduct case studies of two clusters in Section 6.

5.4. The impact of the parameters

To demonstrate how varying the value of the parameters affects LBE identification, we conduct a case study of the Slammer worm incident, mainly because it generates a large amount of LBEs so it is easy to observe the change in trend. The impact of θ_d and the length of time slot is shown in Fig. 9, where we demonstrate the trend of the identified LBEs in accordance to different parameter settings. Unless otherwise specified, $\theta_d = 0.8$, $\theta_w = 40\%$, $\theta_s = 0.7\%$, and the length of time slot is 20 minutes.

It is evident in the figures that minor changes to θ_d and the time slot length do not affect the size trend of identified LBEs. In other words, despite changes in the parameter values, large LBEs remain large, small LBEs remain small or become undetected. By setting a reasonable size threshold according to new θ_d and slot length, the outcome of our method would be similar to the current one. To summarize, while moderate modification to the parameters slightly affects the quantity and size of identified LBEs, it does not impair the correctness of our scheme.

5.5. Algorithm time consumption

In this part, we investigate the time consumption of the GDA algorithm. In the pre-processing, we delete the rows and columns with '1's less than 11 for the purpose of reducing problem scale (introduced in Section 3.3.3). Although we do not adopt a half-way termination scheme, we ignore the UMVs that have size values less than 90% of θ_s, or have width values less than 90% of θ_w. We run the program on a laptop, which has a 2.4GHz Intel Core i3 CPU, a 4GB RAM, and has Ubuntu 14 installed.

Fig. 10 illustrates the time consumption in seconds as a function of input UVM size. In this figure, 99.5% of the time consumption is less than or equal to 10 seconds. Besides, the distribution complies with our previous analysis, i.e., a close-to-linear correlation between the time complexity of GDA algorithm and the size of the input UVM, as indicated by the straight red line.

6. Case studies

In this section, we conduct case studies on a) the largest LBE, b) the largest cluster, and c) a persistent incident that lasted for 200 minutes. This section is an important extension to our previous works [27,28]. All of the LBEs are in the 2013 data set. The purpose of the case studies is two-fold: in addition to illustrating the high impact of the LBEs, we unveil the cause of the LBEs.

6.1. The largest LBE

As shown in Fig. 4, the largest LBE occurs on February 14th, from 3:50 to 4:10 (GMT). It contains 16047 prefixes and 114 monitors, and has a relative size of 0.032.

6.1.1. Major element identification

Method. First of all, we consider a classifier: if a route passes through AS α, the (monitor, prefix) pair of the route will be captured in an LBE. This is a binary classification and each (monitor, prefix) pair is either labeled as 'captured' or 'not captured' after classification.

It is a common practice to use recall and precision to evaluate the performance of a classifier; we also use them here. We denote the recall and precision of the 'captured' label as *Recall*(α) and *Precision*(α), and the two terms are defined as follows. a) *Recall.* We denote the total number of routes that changed and are recorded in LBE after an incident as n_t, and among the n_t routes, n_α went through AS α before the incident, then $Recall(\alpha) = n_\alpha / n_t$. b) *Precision.* We denote the total number of routes that go through AS α before an incident as m_α, and n_α of these routes changed and are recorded in LBE after the incident, then $Precision(\alpha) = n_\alpha / m_\alpha$. One can view n_t as total true number, n_α as true positive number, and m_α as total positive number.

We define the *major element* of an LBE as an AS (or AS link) such that a) most of the routes captured by LBE went through it, and b) most of the routes went through it are captured by LBE. Condition a) means *Recall*(α) is high and condition b) means *Precision*(α) is high. In this part we focus mainly on AS, and analyze AS links when necessary.

Note that the major element of an LBE is not necessarily the cause of the underlying incident. For example, for a path change, the cause could be an AS that exists only in the path after the change [6], and it could even be an AS in neither of the paths before and after the change [45]. So we just use the identified major element (if exists) as a starting point to examine the incident.

Data and result. To obtain recall, we download the routing tables of the Routeviews and RIPE collectors to get the routes before the incident. Unlike updates, which have accurate time stamps, routing tables are collected every eight hours. So we get thetables before and closest to the LBE in time, in order to obtain a snapshot of the routes just before the incident. We omit the tables with too small size, i.e., smaller than 90% of the average size within the month, even if they are the best in terms of time.

We extract 1819660 routes from the routing tables with regard to the monitors and prefixes in the LBE. Note that not all of these monitors are able to see all of these prefixes. After deleting the redundant AS numbers in the AS paths of the routes, we get the recall value of each involved AS.

To obtain precision, we get totally 49940122 routes from the routing tables with regard to the monitors in the LBE. The top ten

Table 1
The top 10 ASs with the highest recall values.

Monitor AS (AS number)	Recall	Precision
CenturyLink (209)	0.8776	0.8784
DoD Network Information Center (721)	0.3942	0.9971
DoD Network Information Center (27064)	0.2703	0.9989
Level 3 Communications (3356)	0.2366	0.0399
CenturyTel (22561)	0.177	0.9853
Cogent Communications (174)	0.1115	0.0387
NTT Communications (2914)	0.0987	0.0406
Global Telecom & Technology (3257)	0.0955	0.0348
Level 3 Communications (3549)	0.0893	0.03
TeliaSonera International Carrier (1299)	0.0871	0.0387

ASs with the highest recall values are listed in Table 1; the precision values of these ASs are also presented.

Analysis. In the table, seven out of the ten ASs are tier-1 ASs; the exceptions are AS721, AS27064, and AS22561. CenturyLink has the highest overall recall and precision. Even though the precision values of AS721 and AS2764 are higher than that of CenturyLink (explained later), their recall values are much smaller.

Through analyzing the topological relationship of these ASs, we find that all of the three non-tier-1 ASs are in the customer cone of CenturyLink. Specifically, AS721 connects to the Internet mainly through CenturyLink; AS27064 connects to the Internet through the AS721-AS209 link; AS 22561 connects to the Internet through only CenturyLink. Note that all of the three non-tier-1 ASs have very high precision values.

Actually, we also observe other ASs with high precision values (not shown in the table), e.g., AS27065 (precision=0.9991), AS27066 (precision=0.9815), and AS5972 (precision=1.0). The majority of these ASs are in the customer cone of CenturyLink, indicating the critical role of CenturyLink in the incident.

To further understand the incident, next we analyze the content of the updates.

6.1.2. Update pattern

The total update quantity in this LBE is 5864806; to understand the information carried in these updates, we categorize update pattern into eight types, listed in Table 2. The quantity of each type is shown in Table 3.

The result indicates that the main impact of the incident is a large number of path and policy changes. Besides, the high amount of identical updates (AADup1) indicates induced pathological behaviors. AADup1 is mainly due to the interaction between iBGP and eBGP in the monitor AS [46], e.g, Next-hop and Cluster-list changes in iBGP may lead to identical eBGP updates. Since we have no iBGP information, we focus on only AADiff+WADiff and AADup2+WADup2.

6.1.3. Analysis: path changes

The total quantity of AADiff and WADiff is 1797396. For each changed path, we divide the involved ASs into three segments: a) *stable-segment*, which remains unchanged; b) *before-segment*, which changes; c) *after-segment*, which is the one that a before-segment changes to. For each participating AS, we record its number of being in each type of segment. According to the result, AS209 is in 1385591 stable segments, which is the highest, indicating that CenturyLink, as the major element, is not the changing element per se. Furthermore, the ASs in the customer cone of AS209 (e.g., those mentioned in Section 6.1.1) are in much more stable segments that in other types of segments.

Next, we record the number of each (before-segment, after-segment) pair. Note that since a path change may be due to the underlying incident or the BGP convergence process, and temporary path changes are common in the BGP system, we focus on only the most prevalent characteristic of the path changes. We find that most of the changing elements are tier-1 ASs. Specifically, 46.88% (i.e., 842633) of the change pairs contain only tier-1 AS in both segments, and 79.22% (i.e., 1423943) of the change pairs contain only tier-1 AS in at least one segment.

The analysis indicates that the peering links between AS209 and most other tier-1 ASs are very unstable in this incident. Such instabilities affect most of the prefixes that are reachable through AS209.

6.1.4. Analysis: BGP Community changes

The total quantity of AADup2 and WADup2 is 1235528, and 98.58% of them are community changes. We omit the other types of change, e.g., change of the origin attribute.

We record the number of each changed community segment pair. We find that the community changes are mostly due to four tier-1 ASs, i.e., AS3257, AS3356, AS2914, and AS3549. Specifically, 71.8% (i.e., 874420) of the changes are community changes of the four ASs.

The semantics of community values for AS3257, AS3356, and AS2914 are publicly available; by checking the semantics, we find that the majority of the community changes indicate location change. For example, the change ‘3356: 2011 $\rightarrow$ 3356: 2022’ informs that the location where the route is learned changes from San Jose to Miami. For AS3257, there are also ‘Private Interconnect ingress tagging’ changes and ‘Receiving router loopback ip address’ changes, which yield similar information as location change does. However, we are unable to identify any pattern in these changes, e.g., coordinated changes from one location to another. Instead, we observe many back-and-forth changes between two locations. We illustrate the fluctuation characteristic of the incident in the next part.

It is well known that tier-1 ASs peer with each other to guarantee connectivity, and a pair of peering tier-1 ASs have BGP sessions at multiple topological or geographical locations to tolerate fault. The result indicates that the changes are mainly due to the connection problem between AS209 and the other tier-1 ASs, and the observation complies with the one in the path change analysis. Note that not all ASs leverage BGP communities to explicitly inform connection/location changes.

6.1.5. Temporary oscillation or intended route change

An important question is, whether the instabilities in the links between CenturyLink and the other tier-1 ASs are due to temporary oscillation or intended route change. To answer it, we analyze the routes before and after the incident. Specifically, for each (prefix, monitor) pair in the LBE, we get its route from the routing tables before the incident, and compare it with its last update (if exists) within the period from 3:50 to 4:30. Note that we append 20 minutes to the end of the LBE in order to capture possible missed instabilities in the incident.

We record four types of change, i.e., a) AS path change, b) community change, c) prefix-withdrawn, and d) no change of the previous three types. For each monitor, we get the ratio of prefixes in each category, and we omit the prefixes without observed updates.

The result is presented in Fig. 11; the x-axis is the monitors ordered by the quantity of ‘No change’ type, and the y-axis is the number of prefixes in each category. Since not every monitor can see all of the affected prefixes in updates, the total number of prefixes for each monitor could be different. As is evident in the figure, the routes of most of the updated prefixes recovered to their original state. Specifically, 91 monitors observe more than 80% of their updated prefixes recover to their original routes. This result indicates that the underlying incident is a temporary oscillation instead of an intended permanent change.

Table 2
Update pattern and description.

Pattern	Description
AW	Withdrawal after announcement to the same IP prefix; could be normal or pathological.
WWDup	Duplicate withdrawals to the same IP prefix; pathological.
WADup1	Duplicate announcement after withdrawal to the same IP prefix; due to transient topological changes or pathological oscillation.
WADup2	Duplicate announcement after withdrawal to the same IP prefix; due to transient policy changes or pathological oscillation.
WADiff	Announcement (different AS path or next hop) after explicit withdrawal to the same IP prefix.
AADup1	Duplicate (identical) announcements to the same IP prefix; pathological because BGP is not supposed to send repeated announcements.
AADup2	Duplicate (same AS path and next hop, but not identical) announcements to the same IP prefix; mainly due to routing policy changes.
AADiff	New-path announcement after announcement to the same IP prefix; implicit withdrawal due to route changes.

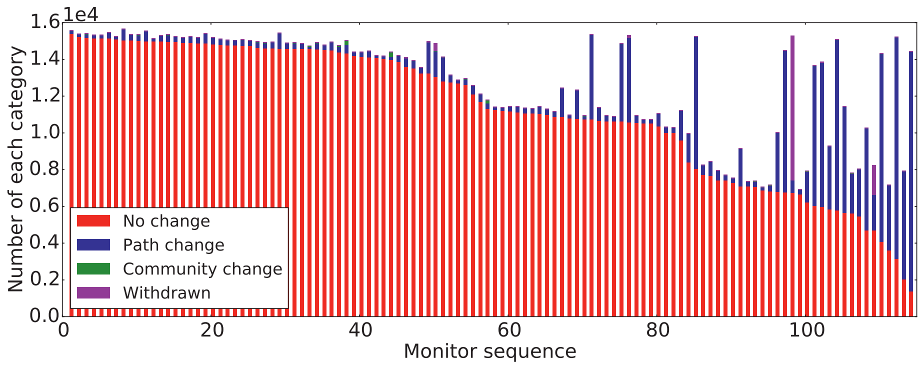


Fig. 11. Comparison between the route in the routing table and that in the last update.

Table 3
Quantity of each update pattern.

Pattern	number	Pattern	number
AADiff	1565265	AADup2	1157912
AADup1	605679	AW	409200
WADiff	232131	WADup2	77616
WADup1	75988	WW	140

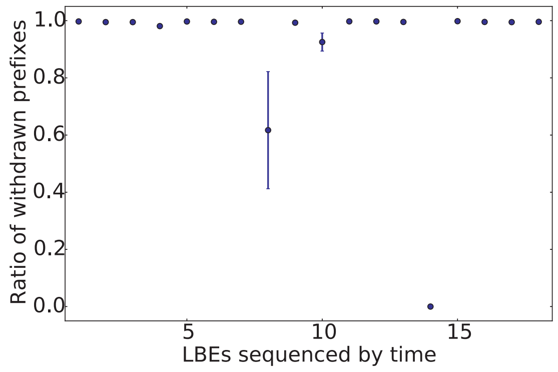


Fig. 12. The ratio (mean +- variance) of withdrawn target prefixes.

According to the analysis, we speculate the incident is due to the temporary instability between CenturyLink and the other tier-1 ASs. Most of the prefixes reachable through CenturyLink are affected, and most tier-1 ASs are involved in the incident. The incident is caused by CenturyLink; it could be a configuration error, or temporary failure in certain facilities. The incident does not look like an intended rerouting because most of the affected routes recovered to their original states.

6.2. The largest cluster

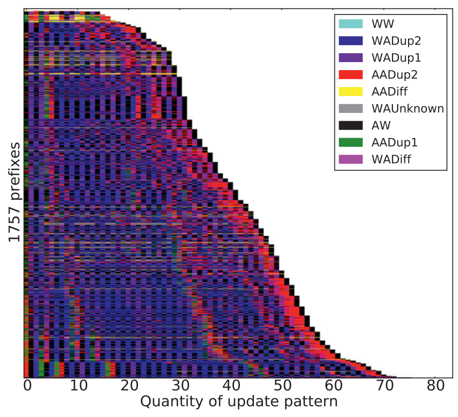
The largest cluster in the 2013 data set contains 18 LBEs, and the common prefix set of these LBEs contains 1328 prefixes. In our previous work [28], we presented some very preliminary analysis of these LBEs, lacking both major player identification and a full picture of their impact. We believe the analysis presented in this part is much more comprehensive.

13 of these LBEs are in April (part one) and 5 are in June (part two). The time span of part one is 1400 minutes (from 10 April 07:30 to 11 April 06:50, GMT), and that of part two is 120 minutes (20 June from 17:10 to 19:10, GMT). The time pattern shows that none of these LBEs is correlated with its previous or next slot, indicating that the underlying incident is a series of intermittent sub-incidents instead of a continuous one.

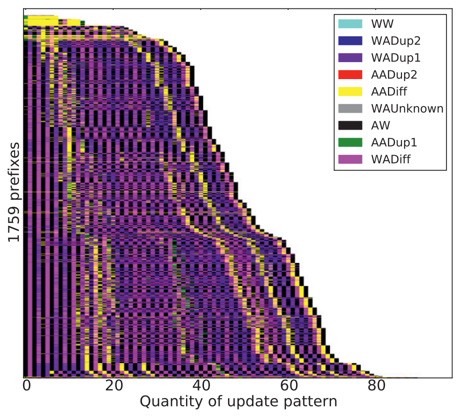
Like what we've done previously, we attempt to get the major element of these LBEs. We illustrate the analysis of the first LBE in part one; the results for the other LBEs are similar. For this LBE, we cannot identify a major element, which is supposed to have both high recall and precision. Specifically, the top-3 ASs with the highest recall values are AS3356 (recall=0.2062), AS26496 (recall=0.1551), and AS3549 (recall=0.1226), and their precision values are 0.0054, 0.6608, and 0.0065, respectively.

Further investigation shows that an important reason for not identifying a major element is that we could not get the routes of as many as 1773 prefixes in the routing tables before the incident. In the updates, the AS paths of these prefixes end with AS47331. Through longest-prefix-matching in the routing tables, we attribute these prefixes to more than 100 prefixes owned by AS9121. In terms of topology, AS47331 connects to the Internet through only AS9121, and the two ASs belong to the same organization. Therefore, the announcement of the 1773 prefixes is due to AS9121 and AS47331. AS9121, as the only provider of AS47331, fails to filter out the excessively large amount of announced prefixes that originally belong to itself.

Next, we investigate the content of updates to further understand the incident. For the first LBE, we find that almost all of the aforementioned 1773 prefixes are withdrawn in the same time slot. Similar phenomena occur in other LBEs in the cluster. As shown in Fig. 12, the x-axis is the series of LBEs in the cluster sequenced by time. For each LBE, we select the *target prefixes* as those originated



(a) Monitor IP: 4.69.184.193



(b) Monitor IP: 199.87.240.1

Fig. 13. Examples of update pattern series.

by AS47331 (in updates) but are attributed to AS9121 through longest-prefix-matching, because the instabilities related to these prefixes are the major impact of the incident. The dots show the mean ratio of withdrawn target prefixes for all the monitors in the corresponding LBE, and the error bars (some are very close to the dots) denote the variance.

As is evident from the figure, in most of these LBEs we observe withdrawals to the vast majority of the target prefixes, except for the 8th and 14th LBEs. Both of the two LBEs are closely followed by another LBEs. In such cases, one cannot divide the sub-incidents by the slot border; it could be that the prefixes are announced in the former LBE, and are withdrawn in the latter LBE.

To summarize, the incident is likely due to the configuration error at AS47331 and AS9121. We believe the anomalous behaviour is not intended, because the resulting instability mainly affects the connectivity and performance from other part of the Internet to the networks owned by AS9121 and AS47331; in other words, AS9121 and AS47331 themselves are the major victims of the anomaly.

6.3. A persistent incident

Now we turn to the third largest cluster in the 2013 data set. This cluster contains four LBEs, and the gaps between these LBEs are 20, 0, and 60 minutes respectively. The time pattern of these LBEs shows that the underlying incident is continuous during from 08:10 to 11:30 on September 11, 2013; the duration is 200 minutes. Within this period, 17191459 updates are observed. The common prefix set of the LBEs contains 1769 prefixes, and we can find the routes for only 803 of them in the routing tables right before the incident. Except for several cases, all of the prefixes are withdrawn at the end of the incident. Next, we turn to the content of updates to obtain information about the 1769 prefixes.

Unlike the largest cluster, we could not identify a dominating origin AS for these prefixes. Instead, we observe in the corresponding updates two intermediate ASs that exist in almost all (with only several exceptions) the paths to the 1769 prefixes, and the observation is consistent among the 127 monitors in the common monitor set. Specifically, AS28207 announces these prefixes to AS3549, which is a tier-1 AS, and there is one or more hops between AS28207 and the origin ASs of the prefixes.

However, among the 803 prefixes having routes in the routing tables, only 32.5% and 41.62% of their routes pass through AS28207 and AS3549, respectively. For the other 966 prefixes, we get their less-specific prefixes hence the routes by longest-prefix-matching in the routing tables, and find that only 0.16% and 36.71% of these routes go through AS28207 and AS3549. Therefore, the incident is either AS path spoofing or route leakage by AS28207.

A question is: why the incident lasts for so long? The answer is the multiple rounds of announcement-withdrawal cycle of these prefixes. Specifically, the number of the major update patterns is as follows; AW: 5341608, AADiff: 3458788, AADup2: 2718339, AADup1: 290773. In a more-detailed analysis of each monitor, we find that withdrawals are so frequent that in most cases there are few updates between two successive withdrawals. We present two examples in Fig. 13a and b. The x-axis is the number of update pattern, and the y-axis is the prefixes (we omit the prefixes that have too many updates thus make the figures unreadable). Due to different topological locations, while the former monitor observes mostly AW and WADup2, the latter monitor observes mostly AW and WADiff. Both figures show multiple announcement-withdrawal cycles.

To summarize, AS28207 incorrectly announces more than 1000 prefixes that does not belong to or go through this network. The duration of the incident is long and the impact of the incident is significant; the incident is either an intended path spoofing or an unintended route leakage.

7. Conclusion

Most of the traditional works on detecting and analyzing anomalies in the BGP system are prone to the artifact related to monitor-local events. To cope with the issue, we propose the concept of Update Visibility Matrix (UVM) and Large-scale BGP Event (LBE). We formulate the problem of identifying LBE in UVM, then propose an algorithm to solve it. Our method explicitly avoids the detection of monitor-local and low-impact events. We apply the method to the updates related to twelve famous incidents and observe a strong correlation between the incidents and the identified LBEs. We also analyze ten months' data in 2013 and identify 101 LBEs that have never been detected and investigated before. The analysis of these identified LBEs validates the effectiveness of our method and basic idea. Finally, we conduct case studies on three incidents in the 2013 data set, which involve 23 LBEs. The detailed examination shows the high impact and cause of these incidents, which further enhance the importance of our work. The measurement results suggest that our study could be helpful in network/Internet operation, management, and monitoring tasks.

Acknowledgment

The research is supported by the National Basic Research Program of China (973 Program) under Grant 2012CB315803 and

2012CB315806, the National Natural Science Foundation of China under Grant 61133015, the National Natural Science Foundation of China under Grant 61502268 and 61402255, the National High-Tech Research and Development Program of China (863 Program) under Grant 2015AA015701, Specialized Research Fund for the Doctoral Program of Higher Education under Grant 20120002110060.

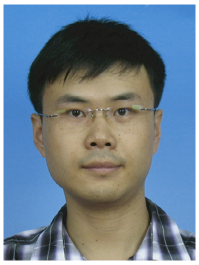
References

[1] J. Li, S. Brooks, I-seismograph: observing and measuring internet earthquakes, in: IEEE INFOCOM, 2011, pp. 2624–2632.
[2] Y. Liu, X. Luo, R. Chang, J. Su, Characterizing inter-domain rerouting by betweenness centrality after disruptive events, IEEE JSAC 31 (6) (2013) 1147–1157.
[3] G. Comarela, M. Crovella, Identifying and analyzing high impact routing events with PathMiner, in: ACM IMC, 2014, pp. 421–434.
[4] S. Deshpande, M. Thottan, T. Ho, B. Sikdar, An online mechanism for BGP instability detection and analysis, IEEE Trans. Comput. (11) (2009) 1470–1484.
[5] J. Mai, L. Yuan, C. Chuah, Detecting BGP anomalies with wavelet, in: NOMS, 2008, pp. 465–472.
[6] A. Feldmann, O. Maennel, M. Mao, A. Berger, B. Maggs, Locating internet routing instabilities, in: ACM SIGCOMM, 2004, pp. 205–218.
[7] J. Rexford, J. Wang, Z. Xiao, Y. Zhang, BGP routing stability of popular destinations, in: ACM SIGCOMM Workshop on Internet Measurment, 2002, pp. 197–202.
[8] J. Wu, M. Mao, J. Rexford, J. Wang, Finding a needle in a haystack: pinpointing significant BGP routing changes in an IP network, in: NSDI, 2005, pp. 1–14.
[9] A. Sapegin, S. Uhlig, On the extent of correlation in BGP updates in the Internet and what it tells us about locality of BGP routing events, Comput. Commun. 36 (2013) 1592–1605.
[10] C. Labovitz, R. Malan, F. Jahanian, Internet routing instability, in: ACM SIGCOMM, 1997, pp. 115–126.
[11] J. Li, M. Guidero, Z. Wu, E. Purpus, T. Ehrenkranz, BGP routing dynamics revisited, SIGCOMM CCR 37 (2) (2007) 5–16.
[12] K. Zhang, A. Yen, X. Zhao, D. Massey, F. Wu, L. Zhang, On detection of anomalous routing dynamics in BGP, in: Networking, Springer Berlin Heidelberg, 2004, pp. 259–270.
[13] V. Menon, W. Pottenger, A higher order collective classifier for detecting and classifying network events, in: IEEE International Conference on Intelligence and Security Informatics, 2009, pp. 125–130.
[14] R. Oliveira, R. Izhak-Ratzin, B. Zhang, L. Zhang, Measurement of highly active prefixes in bgp, in: GLOBECOM, 2005, pp. 894–898.
[15] S. Teoh, K. Zhang, S. Tseng, K. Ma, F. Wu, Combining visual and automated data mining for near-real-time anomaly detection and analysis in BGP, in: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, 2004, pp. 35–44.
[16] W. Liang, Y. Li, J. Bi, G. Zhang, On the accurate identification of familiar inter-domain routing instabilities, in: IEEE GLOBECOM, 2008, pp. 1–6.
[17] The BGPmon Project, (http://www.bgpmon.io/). Accessed: July 11, 2016.
[18] X. Shi, Y. Xiang, Z. Wang, X. Yin, J. Wu, Detecting prefix hijackings in the internet with argus, in: ACM IMC, 2012, pp. 15–28.
[19] R. Mahajan, D. Wetherall, T. Anderson, Understanding BGP misconfiguration, in: ACM SIGCOMM, 2002, pp. 3–16.
[20] M. Lad, X. Zhao, B. Zhang, D. Massey, L. Zhang, Analysis of BGP update surge during slammer worm attack, in: Distributed Computing - IWDC, 2918, Springer Berlin Heidelberg, 2003, pp. 66–79.
[21] A. Dainotti, C. Squarcella, E. Aben, K. Claffy, M. Chiesa, M. Russo, A. Pescapé, Analysis of country-wide internet outages caused by censorship, in: ACM IMC, 2011, pp. 1–18.
[22] Y. Liu, X. Luo, R. Chang, J. Su, Characterizing inter-domain rerouting after japan earthquake, in: IFIP Networking, 2012, pp. 124–135.
[23] C. Zou, W. Gong, D. Towsley, L. Gao, The monitoring and early detection of internet worms, IEEE/ACM TON 13 (5) (2005) 961–974.
[24] M. Siddiqui, D. Montero, R. Serral-Gracia, M. Yannuzzi, Self-reliant detection of route leaks in inter-domain routing, Comput. Netw. 82 (2015) 135–155.
[25] M. Ganiz, S. Kanitkar, M. Chuah, W. Pottenger, Detection of interdomain routing anomalies based on higher-order path analysis, in: Sixth International Conference on Data Mining, 2006, pp. 874–879.
[26] X. Dimitropoulos, G. Riley, Efficient large-scale BGP simulations, Comput. Netw. 50 (12) (2006) 2013–2027.
[27] M. Chen, M. Xu, X. Song, Y. Yang, Towards identifying Large-scale BGP Events, in: IEEE Conference on Local Computer Networks (LCN), 2015, pp. 165–168.
[28] M. Chen, M. Xu, Q. Li, X. Song, Y. Yang, Detect and analyze Large-scale BGP events by bi-clustering update visibility matrix, in: IEEE International Performance Computing and Communications Conference (IPCCC), 2015, pp. 1–8.
[29] I. Bomze, M. Budinich, P. Pardalos, M. Pelillo, The maximum clique problem, in: Handbook of Combinatorial Optimization, Springer US, 1999, pp. 1–74.
[30] Route Views Project, (http://www.routeviews.org/). Accessed: November 2 2014.
[31] RIPE RIS Raw Data, (http://www.ripe.net/data-tools/stats/ris/ris-raw-data). Accessed: November 2 2014.
[32] Active BGP entries of AS 65000, (http://bgp.potaroo.net/as2.0/bgp-active.html). Accessed: November 19 2014.
[33] R. Oliveira, D. Pei, W. Willinger, B. Zhang, L. Zhang, The (in)completeness of the observed Internet AS-level structure, IEEE/ACM TON 18 (1) (2010) 109–122.
[34] Customer cones of Autonomous Systems, (http://data.caida.org/datasets/2013-asrank-data-supplement/data/). Accessed: January 22 2015.
[35] AS to nation/district mapping, (http://www.cidr-report.org/as2.0/autnums.html). Accessed: January 23 2015.
[36] P. Cheng, B. Zhang, D. Massey, L. Zhang, Identifying BGP routing table transfers, Comput. Netw. 55 (3) (2011) 636–649.
[37] A. Sahoo, K. Kant, P. Mohapatra, Characterization of BGP recovery time under large-scale failures, in: IEEE ICC, 2006, pp. 949–954.
[38] A. Sahoo, K. Kant, P. Mohapatra, BGP convergence delay after multiple simultaneous router failures: characterization and solutions, Comput. Commun. 32 (7) (2009) 1207–1218.
[39] C. Labovitz, A. Ahuja, A. Bose, F. Jahanian, Delayed internet routing convergence, in: ACM SIGCOMM, 2000, pp. 175–187.
[40] A. Feldmann, H. Kong, O. Maennel, A. Tudor, Measuring BGP pass-through times, in: Passive and Active Network Measurement, in: Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 267–277.
[41] E. Chan, X. Luo, W. Fok, W. Li, R. Chang, Non-cooperative diagnosis of submarine cable faults, in: Passive and Active Measurement Conference, 2011, pp. 224–234.
[42] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: ACM KDD, 1996, pp. 226–231.
[43] S. Guha, R. Rastogi, K. Shim, Cure: an efficient clustering algorithm for large databases, in: ACM SIGMOD Record, 27, 1998, pp. 73–84.
[44] B. Zhang, S. Srihari, Fast k-nearest neighbor classification using cluster-based trees, IEEE . Pattern Anal. Machine. Intell. 26 (4) (2004) 525–528.
[45] U. Javed, I. Cunha, D. Choffnes, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, PoiRoot: investigating the root cause of interdomain path changes, in: ACM SIGCOMM, 2013, pp. 183–194.
[46] J. Park, D. Jen, M. Lad, S. Amante, D. McPherson, L. Zhang, Investigating occurrence of duplicate updates in BGP announcements, in: Passive and Active Measurement Conference, 2010, pp. 11–20.

Meng Chen received the B.S. degree (2011) from Beijing Normal University. He is now a Ph.D candidate in the Department of Computer Science & Technology at Tsinghua University. He was a visiting Ph.D. student at the Hong Kong Polytechnic University in 2014. His major research interests include routing protocol and Internet measurement.

Mingwei Xu received the B.Sc. degree and the Ph.D. degree from Tsinghua University. He is a full professor in Department of Computer Science at Tsinghua University. His research interest includes computer network architecture, high-speed router architecture and network security. He is a member of the IEEE.

Qing Li received the B.S. degree (2008) from Dalian University of Technology, Dalian, China, the Ph.D. degree (2013) from Tsinghua University, Beijing, China; all in computer science and technology. He is currently an assistant researcher in the Graduate School at Shenzhen, Tsinghua University. His research interests include reliable and scalable routing of the Internet, software defined networks and information centric networks.



Yuan Yang received the B.Sc., M.Sc., and Ph.D. degree from Tsinghua University. He is now a postdoc. in the Department of Computer Science & Technology at Tsinghua University. He was a visiting Ph.D. student at the Hong Kong Polytechnic University between 2012 and 2013. His major research interests include computer network architecture, routing protocol, and green networking. He is a member of the IEEE.