An Online Mechanism for BGP Instability Detection and Analysis

Shivani Deshpande, Marina Thottan, Tin Kam Ho and Biplab Sikdar *Member, IEEE*

Abstract—The importance of Border Gateway Protocol (BGP) as the primary inter-Autonomous System (AS) routing protocol that maintains the connectivity of the Internet imposes stringent stability requirements on its route selection process. Accidental and malicious activities such as misconfigurations, failures and worm attacks can induce severe BGP instabilities leading to data loss, extensive delays and loss of connectivity. In this work we propose an online instability detection architecture that can be implemented by individual routers. We use statistical pattern recognition techniques for detecting the instabilities and the algorithm is evaluated using real Internet data for a diverse set of events including misconfiguration, node failures and several worm attacks. The proposed scheme is based on adaptive segmentation of feature traces extracted from BGP update messages and exploiting the temporal and spatial correlations in the traces for robust detection of the instability events. Furthermore, we use route change information to pinpoint the culprit ASes where the instabilities have originated.

Index Terms—BGP, Anomaly Detection, Routing Instability, Statistical Pattern Recognition

1 Introduction

With BGP being the default inter-domain routing protocol used in the Internet today, its stability and robustness is critical for ensuring the delivery of packets and maintaining connectivity. Since BGP is primarily responsible for routing traffic across different administrative domains, its route selection mechanisms are governed by the policies of these domains. Therefore, the message handling procedures of BGP are designed to support such policy-based route selection. This intrinsic property of BGP magnifies any route change event in the Internet with an elaborate path exploration process. Under stressful conditions like worm attacks, link outages or router failures, this behavior of BGP causes severe route fluctuations [4], [11] leading to loss of connectivity for several networks for prolonged periods of time. These instabilities also result in widespread degradation of the network's end-to-end utility and the user's perception of the quality of service offered by the Internet. It is thus necessary to be able to detect and limit the impact of routing instabilities. In this paper, we propose an architecture for an online detection scheme that uses statistical patterns in the BGP update message data to detect instability events and isolate the location where the event originated.

There has been some recent work focusing on the detection of routing anomalies using BGP update message data. Tools for visualization-aided anomaly detection and root-cause analysis of BGP anomalies are presented in [20], [25] and [19] respectively. An instance-learning based scheme using wavelets for detecting BGP anomalies is proposed in [28]. In [26] the authors use BGP data from multiple border routers in a single AS along with traffic load information inside the AS to detect BGP routing changes that impact large amounts of traffic. The authors of [9], [27] use principal component analysis (PCA) based schemes to identify network disruptions and for root cause analysis. A combination of routing update analysis and data plane fingerprinting is used in [8] to detect IP prefix hijacking. A data plane based IP prefix hijacking scheme requiring multiple observation points is proposed in [30] while the scheme in [29] is based on active probing from multiple networks. The system proposed in [12] fails to detect hijacks that occur without a change in the origin AS and may not always distinguish valid changes from actual hijacks.

In contrast to existing work, we propose an online system for detecting the onset of instabilities in BGP and isolate the location of its origin, without the need for human intervention. The proposed system can be implemented on a single router without raising issues concerning widespread deployment and modification of standards. Our work uses statistical techniques to detect instabilities using the time domain characteristics of BGP update message data and does not need any additional probing. We first identify and extract features from the BGP update message data that have a temporal correlation with instability events. Next, we apply filtering and adaptive segmentation techniques on the time-series fea-

- *S. Deshpande now works at BlueCoat Systems, Sunnyvale, CA, USA. The work was done while she was at Rensselaer Polytechnic Institute, Troy, NY, USA.*
  E-mail:stephenson@mitre.org
- *M. Thottan is with the Center for Networking Research at Bell Laboratories, Alcatel-Lucent, Murray Hill, NJ, USA.*
  E-mail:marinat@alcatel-lucent.com
- *T. K. Ho is with the Statistics and Learning Research Department of Bell Laboratories, Alcatel-Lucent, Murray Hill, NJ, USA.*
  E-mail:tkh@research.bell-labs.com
- *B. Sikdar is with the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA.*
  E-mail:sikdab@rpi.edu

Manuscript received June 7, 2008; revised December 5, 2008.
This work supported in part by NSF grant 0347623.

ture data to isolate periods of instabilities. Correlations in the presence of abnormalities across several features are used to reduce the occurrence of false positives in the detection. Our approach is thus mixed: it is based on pure statistical techniques but the underlying data used provides a more meaningful insight into the cause and effect of the instabilities.

The main contributions of this paper are the following: **(1)** we introduce new features extracted from BGP message data that capture the topological changes induced by instabilities and show distinct behavioral changes during periods of instability; **(2)** we characterize the BGP instability behavior observable at any router using statistical methods and show that the characterization is effective at detecting anomalous events; **(3)** we develop a mechanism to isolate the ASes where the instabilities originate from; **(4)** we evaluate the performance of our schemes using real BGP data. The proposed method performs uniformly well under all kinds of instabilities and across different topologies and policies and is easy to deploy.

The rest of the paper is organized as follows: Section 2 presents an overview of the proposed architecture. Section 3 describes the features extracted from BGP messages, Section 4 explains the detection scheme, Section 5 describes the root cause location isolation process and Section 6 presents the parameter estimation process. Section 7 evaluates the proposed mechanism's performance. Finally, Section 8 presents the concluding remarks.

2 System Architecture

This section presents a description of the architecture of the proposed system and its components and their functions. The proposed system can be broadly described as a two part scheme: the first part detects the BGP instability events while the second pinpoints the source of the instability to allow possible correcting actions. Figure 1 shows the various components of the proposed system and the interactions between them. We emphasize that this architecture is implemented entirely on a single BGP router and the proposed functionalities add very little to the computational load on the router.

The input data used by the proposed mechanism are BGP update messages received by any BGP router from its peers as part of its routing operations. The time series features extracted from the update message data let us view this as an anomaly detection problem. We now describe the functions performed by the different blocks.

Feature Extractor: It performs the basic function of parsing the needed features from the BGP update messages received by a router from its peers. It separates the update messages received from different peers into different datasets and then performs the necessary parsing on these datasets to obtain the feature traces. The features are extracted on data collected every 5 minutes in order to limit the rate at which data needs to be processed. We describe the features used in Section 3.

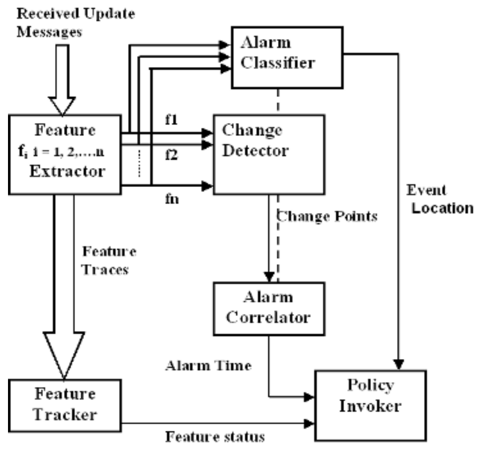


Fig. 1. Overall architecture of the detection and root cause location scheme

Change Detector: The important function of tracking the behavior of the feature time series and detecting when a substantial change occurs in them is performed here. The change detection is performed for every feature trace extracted from each peer separately, in parallel, and the process is described in Section 4.1.

Alarm Correlator: It is used to make the detection robust against feature volatility. It implements the algorithm described in Section 4.2 to establish whether the alarms obtained from the different feature traces for the same peer are correlated. The correlation is checked across different combinations of features using a decision tree based mechanism. These correlated alarms, if obtained, are termed per-peer alarms. The per-peer alarms can be further tested for spatial correlation to determine the extent of the detected instability.

Alarm Classifier: It helps in identifying the location of occurrence of the root cause event that caused the instabilities. It uses the AS-path data for this purpose.

Policy Invoker: It invokes control policies that help prevent route fluctuations caused by the instability from propagating to the downstream peers of this BGP router. This block is optional and the router can be configured to only raise an alarm to signal the occurrence of an instability. The development of control policies is outside the scope of this paper.

Feature Tracker: This block is used together with the policy controller. It monitors the behavior of the features for continued presence of anomalous symptoms after the detection of an instability. Once the features appear to return to normality the policy controller is notified and the route selection process is switched back to normal. The feature tracker can be a simple threshold-based mechanism which tracks the levels of the different features being used. It implements a timer based mechanism to readjust its thresholds to guard against a permanent policy switch after a false positive.

3 Feature Selection

When a route or node failure occurs, each BGP router initiates a path exploration process for the failed routes.

The path exploration process involves the exchange of route withdrawal and announcement messages between BGP routers. These update messages show certain characteristics that are specific to periods of instability and form the features used by our instability detection mechanism.

From the BGP update messages received by a router, we identify and extract *features* that are used to differentiate between BGP's behavior during normal and anomalous periods. The features are collected separately for each peer of a router and are used in the form of a time series collected every 5 minutes. The feature traces are then median filtered to smooth out any unwanted transients and the figures shown in this section show these filtered traces. The features were identified using a scatter plot of the data traces in the Mirage software tool [7]. We now describe these features in detail.

3.1 AS Path Length

Routing instabilities cause established paths to become unavailable or may result in certain destinations being unreachable. As a result, instabilities are characterized by route withdrawals and the BGP path exploration process to find an alternative route to the same destination. Under such circumstances, we observed the occurrence of many routes with abnormally long AS path lengths, as shown in Figure 2. This is because in the absence of stable paths of shorter lengths, BGP routers try to use longer alternative paths.

Another reason for the receipt of routes with abnormally large AS path lengths during instabilities is the common practice of AS path prepending [22]. AS path prepending is the practice where a BGP router prepends its AS number multiple times consecutively instead of just once to an AS path it advertises. This is done to make the path through it less attractive to the BGP peers that base their route selection on the shortest path length criteria. As a result, these routes are the very rarely used backup paths. During a failure however, these routes are also eventually selected when all other shorter routes fail and as a result, they can form a considerable percentage of the number of AS paths received by a BGP router. Note that the AS path length that we use is just the count of AS numbers listed in the AS path sequence received in the message and are not required to be a unique list of AS numbers.

During normal periods of operation, the lengths of the AS paths advertised by any given peer of a BGP router show only a small deviation around the average value. We call the mode values of the distribution of AS path lengths during normal periods as the "normal value". Thus, the number of messages received with AS path lengths differing from this normal value is relatively small during normal periods of operation but shows a prominent increase under instability conditions, as shown in Figure 2. We use the length of the AS paths received from a peer as one of our features and define

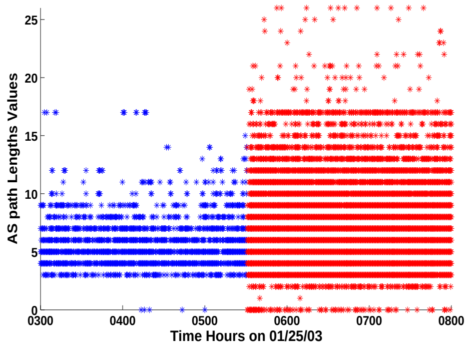


Fig. 2. Number of messages with different AS path lengths received from the router in AS513 during 5 hours around the onset of the Slammer worm attack on 25th January 2003. The red part (right half) indicates the time period after the attack started. As can be seen there is a sudden increase in the number of messages with AS path lengths greater than (4 or 5) which is the normal average at the onset of the worm (midpoint of the interval).

it as

$$ASPL = \{\bar{X}_{ij} = \langle x_0, x_1, \cdots \rangle; i = 1, \cdots, M_l; j = 1, \cdots, NP\}$$

where, $\bar{X}_{ij}$ is a time series of the number of messages with AS path length $= i$, received over every 5 minute interval from peer number j, M_l is the maximum observed AS path length value and NP is the number of peers of the local BGP router.

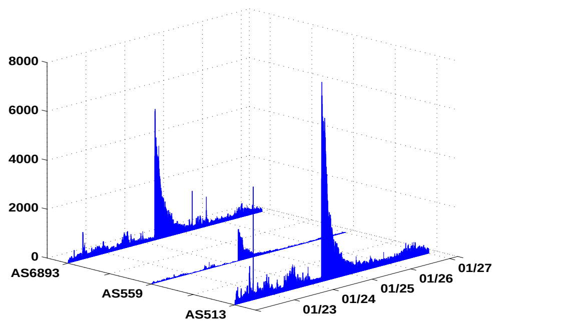
3.2 AS Path Edit Distance

During an instability, not only are a large number of long AS paths exchanged but also a large number of "rare" AS paths are advertised. We quantify the latter effect by treating AS paths received in consecutive messages as strings and obtaining edit distances [21] between them as a measure of their dissimilarity. We define the edit distance between any two AS paths as the minimum amount of AS number substitutions, deletions and insertions (or combinations thereof) needed to convert one path into another. As an example, consider the sequence of messages received at AS2 from AS6 [6-1-7; 6-4-3-1-7]. The edit distance between these AS paths can be counted as 2 insertions. If on the other hand because of a link failure between AS6 and AS7, the path advertised by AS2 to AS3 changes from [2-6-7] to [2-1-7], then the edit distance between the two AS paths will be one substitution.

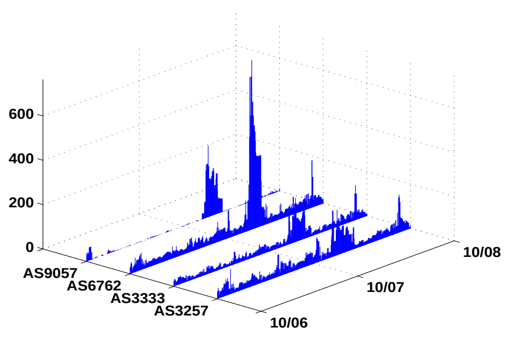
During the path exploration following any instability event, as all possible paths for a particular destination are exchanged, a large number of successive messages show higher edit distances. Fig. 3 shows this effect for the Slammer worm attack. The AS path edit distance feature set is defined as

$$ASPED = \{\bar{X}_{ij} = \langle x_0, x_1, \cdots \rangle; i = 1, \cdots, M_{ed}; j = 1, \cdots, NP\}$$

where, $\bar{X}_{ij}$ is a time series of the number of messages with AS path edit distance i, received over every 5



(a) BGP announcement message volumes from 23rd to 27th of January 2003. The Slammer worm attacked the Internet on the 25th of January.



(b) BGP withdrawal message volumes from 6th to 8th of October 2001. A BGP misconfiguration error occurred on the 7th of October.

Fig. 4. Message volumes for two periods of instability

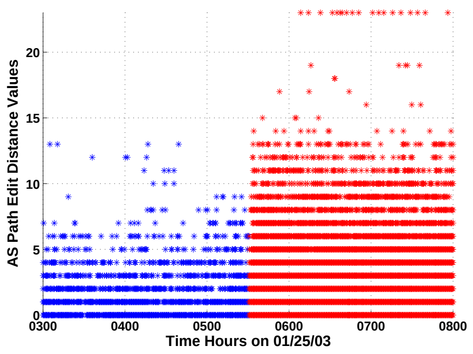


Fig. 3. Number of messages with different pairwise AS path edit distances received from the router in AS513 during 5 hours around the onset of the Slammer worm attack on 25th January 2003. The red part (right half) indicates the time period after the attack started. At the onset of the attack (midpoint of the interval) there is a sudden increase in the number of successive messages with AS path edit distances greater than (0 or 1) which is the normal value.

minute interval from peer number j and M_{ed} is the maximum observed AS path edit distance value.

3.3 Message Volume

Interdomain routing instabilities also exhibit a sharp and sustained increase in the number of announcement and withdrawal messages exchanged by the BGP routers [4], [11], [5]. To illustrate this effect, Figures 4(a) and 4(b) show the BGP update message volume traces around the period of the instabilities associated with the Slammer worm attack on the Internet on the 25th January 2003 [11] and the leakage of private AS numbers from AS2008 and AS3300 due to a router misconfiguration on 7th October 2001 [13]. We observe a large peak in the number of announcement and withdrawal messages received at a router from its peers during the periods of these instabilities. Thus we consider the volume of announcement and withdrawal messages as possible features that can be used to detect instabilities. These features are defined as

$$AV_i = \{\bar{X}_i = \langle x(0), x(1), \cdots \rangle\}, \forall i = 1, 2, \cdots, NP$$
$$WV_i = \{\bar{Y}_i = \langle y(0), y(1), \cdots \rangle\} \forall i = 1, 2, \cdots, NP$$

where $\bar{X}_i$ and $\bar{Y}_i$ are the time series of the number of announcements and withdrawls received in each 5 minute interval from peer number i, respectively.

3.4 Relevant Features

Based on the features above, our available feature set is:

$$F' = [AV, WV, ASPL, ASPED] \quad (1)$$

where, AV and WV are the volume feature sets, $ASPL$ is the AS path length and $ASPED$ the AS path edit distance feature set. Now, the maximum values of AS path length (M_l) and edit distance (M_{ed}) observed can be very high. We first filter out feature traces for values of AS path length very close to M_l and values of AS path edit distance very close to M_{ed}. This is necessary in order to avoid using very sparse feature traces that can be statistically irrelevant. Also, the sum of the number of messages over all AS path lengths in any 5 minute interval will be equal to the number of announcements received in that interval. Hence we also remove the AV feature trace from consideration (since it is a linear combination of some other features in the set).

Next, we use the Fisher score [23] of each feature to dete-rmine its discriminability for detecting the occurrence of an instability. For this purpose we assume that the detection problem is analogous to a discrimination between the normal periods and periods of instability for the BGP sessions. We use data from a period of 10 hours around the incidence of an anomaly for finding the Fisher scores of all the features. With a sampling rate of 5 minutes the feature traces used are thus 120 samples long, with the first 60 samples belonging to the normal class and the next 60 to the abnormal (unstable) class.

To understand how exactly we obtain the Fisher score for each feature consider the (filtered) feature set with N features:

$$F = \{f_i | i = 1, ..., N\} \quad (2)$$

The Fisher score for each feature is obtained as:

$$R_i = \frac{|m_i^1 - m_i^2|}{s^1(f_i) + s^2(f_i)} \tag{3}$$

where m_i^1 and m_i^2 are the mean values of feature i from the datasets corresponding to the normal and abnormal periods respectively and $s^1(f_i)$ and $s^2(f_i)$ are the corresponding feature scatters, i.e.,

$$s^j = \sum_{y \in D_i} (y - m^j)^2. \tag{4}$$

where D_i is the set of samples of feature i and j is the dataset index. The features are then arranged in the descending order of Fisher scores and added to the classifier one by one until

$$(\sum_{j=1}^{j=M} R_j) > \lambda, \tag{5}$$

where λ is a threshold to ensure enough discriminability in the final feature set. Thus, the final feature set is selected by averaging the sums of the feature scores for datasets from multiple instability events and comparing this average against λ. We then use the following modified version of Equation (5) for final feature selection:

$$\frac{1}{D}(\sum_{k=1}^{k=D} \sum_{j=1}^{j=M} R_j^k) > \lambda, \tag{6}$$

where D is the number of datasets considered here as training events for feature selection. We use $D = 9$, which consists of 3 events with datasets for 3 peers each. We empirically set the value of $\lambda = 0.45$ so as to obtain an optimum number of meaningful features [23].

Table 1 presents the scores of the top 10 features averaged over these 9 datasets. For the features under consideration, we obtain an averaged sum of Fisher scores over different kinds of instabilities as 0.495, which is well above the threshold value for λ and thus the features we have selected have adequate discriminability to be used for detection. Thus, this mechanism successfully validates our current feature set. Consequently, at each router, the final feature set F we use has 9 traces:

$$\begin{aligned} F &= [WV, ASPL', ASPED'] \\ ASPL' &= [\bar{X_{ij}}, |i = 3, 6, 7, 8; j = 1, 2, ..., NP]; \\ ASPED' &= [\bar{X_{ij}}, |i = 2, 3, 4, 5; j = 1, 2, ..., NP] \end{aligned}$$

4 Detection of Instabilities

This section describes our detection mechanism in detail. The detection scheme is based on adaptive sequential segmentation. The core of the segmentation is change detection using a Generalized Likelihood Ratio (GLR) based hypothesis test. The segment boundary detection mechanism uses the GLR test to detect change points. This step is followed first by a process to optimize the segment boundary position and then by a clustering mechanism that exploits the spatial and temporal correlations in multiple features to minimize false alarms.

4.1 Change Detection

This section describes the statistical test and the algorithm for detecting and isolating the instance when an instability occurs. We start with the basic GLR test used for instability detection and then follow it with the algorithms that use this test to detect and optimize the position of the change point.

4.1.1 *Generalized Likelihood Ratio Test*

The test is based on capturing statistical changes in the characteristics of the BGP feature traces using a GLR based hypothesis test. Changes are detected by comparing the variance of the residuals obtained from two adjacent windows of data which are referred to as the learning ($L(t)$) and test ($S(t)$) windows. A windowing mechanism is used to convert the feature time series into the two windows of data (of lengths N_L and N_S respectively):

$$L(t) = \{l_1(t), l_2(t), \cdots, l_{N_L}(t)\} \tag{7}$$

$$S(t) = \{l_1(t), l_2(t), \cdots, l_{N_S}(t)\} \tag{8}$$

that are piece-wise stationary. Residuals are obtained by imposing an AR model on the time series data in each of these windows. Any $l_i(t)$ in the equations above can be expressed as $\tilde{l}_i(t) + \mu$ where $\tilde{l}_i(t) = l_i(t) - \mu$ and μ is the mean of the segment $L(t)$. Now, $\tilde{l}_i(t)$ is modeled as an AR process of order ρ with a residual error $\epsilon_i(t)$

$$\epsilon_i(t) = \tilde{l}_i(t) - \sum_{k=1}^{\rho} \alpha_k \tilde{l}_i(t-k) \tag{9}$$

where $\alpha_L = \{\alpha_1, \alpha_2, \cdots, \alpha_\rho\}$ are the AR parameters. Changes are detected by comparing the variance of the residuals obtained from these two adjacent windows of data. Assuming each residual is drawn from an $N(0, \sigma_L^2)$ distribution, the joint likelihood of the residual time series is given by

$$p(\epsilon_{\rho+1}, \cdots, \epsilon_{N_L} | \alpha_1, \cdots, \alpha_\rho) = \left(\frac{1}{\sqrt{2\pi\sigma_L^2}} \right)^{\acute{N}_L} e^{\left(\frac{-\acute{N}_L \hat{\sigma}_L^2}{2\sigma_L^2} \right)}$$

where σ_L^2 is the variance of the segment $L(t)$, $\acute{N}_L = N_L - \rho$ and $\hat{\sigma}_L^2$ is the covariance estimate of σ_L^2. Using a similar expression for the test window $S(t)$, the joint likelihood ν of the two segments $L(t)$ and $S(t)$ is given by

$$\nu = \left(\frac{1}{\sqrt{2\pi\sigma_L^2}} \right)^{\acute{N}_L} \left(\frac{1}{\sqrt{2\pi\sigma_S^2}} \right)^{\acute{N}_S} e^{\left(\frac{-\acute{N}_L \hat{\sigma}_L^2}{2\sigma_L^2} \right)} e^{\left(\frac{-\acute{N}_S \hat{\sigma}_S^2}{2\sigma_S^2} \right)}$$

where σ_S^2 is the variance of the segment $S(t)$, $\acute{N}_S = N_S - \rho$ and $\hat{\sigma}_S^2$ is the covariance estimate of σ_S^2. The expression for ν is a sufficient statistic and is used to perform a binary hypothesis test based on the GLR. Under the hypothesis H_1 implying that a change is

Feature	ASPL=3	ASPED=2	WDS	ASPL=8	ASPL=6	ASPL=7	ASPL=9	ASPED=5	ASPED=4	ASPED=3
Score	0.0823	0.0629	0.0567	0.0508	0.0496	0.0489	0.0476	0.0436	0.0405	0.0388

TABLE 1
The top 10 features and their feature scores in descending order

Algorithm 1 Change detection and boundary position optimization.

```
$L$  minimum initial window size;
$\delta$  GLR threshold;
$d(x, y)$  GLR dist betn. windows $[1, x]$ and $[x+1, y]$;
$s = L$  initialize $s$ as end of first learning
         and beginning of first test window;
while ($sizeof(data) > 0$) do
  while ($d(s, s + L - 1) < \delta$) do
      $s = s + 1$  grow the learning and slide
                   the test window by one sample;
  end while
  $t_D = s + L - 1$  the change detection point;
  $r = t_D - L + 1$  pointer to the beginning of
                 current test window;
  for ( $t_D - L + 2 \leq s \leq t_D$ ) do
    $g_1 = d(s, s + L - 1)$  GLR dist betn. growing
             learning and fixed test windows;
    $g_2 = d(r, s + L - 1)$  GLR dist betn. fixed
             learning and growing test windows;
    if ($g_1 > g_2$) then
        $r = s$  found better boundary position;
    end if
    $s = s + 1$
  end for
  $ropt = r$  optimal boundary position found;
  $data = [data(r) : data(sizeof(data))]$  further seg-
                 mentation on remaining data;
end while
```

observed between the two windows, we have $\alpha_L \neq \alpha_S$ and $\sigma_L^2 \neq \sigma_S^2$, while under H_0 implying that no change has occurred, $\alpha_L = \alpha_S$ and $\sigma_L^2 = \sigma_S^2$.

Using maximum likelihood estimates for the variance terms, the likelihood ratio is given by

$$\eta = \hat{\sigma}_P^{(\acute{N}_L + \acute{N}_S)} \hat{\sigma}_L^{-\acute{N}_L} \hat{\sigma}_S^{-\acute{N}_S}. \tag{10}$$

where $\hat{\sigma}_P^2$ is the pooled variance of the learning and test windows. For computation purposes we use the logarithmic form of the GLR as distance:

$$d = (\acute{N}_L + \acute{N}_S) \log(\sigma_P) - ((\acute{N}_L \sigma_L) + (\acute{N}_S \sigma_S)) \tag{11}$$

4.1.2 Segment Boundary Detection

The segment boundary detection process isolates periods of abnormal behavior in the feature traces. The boundary/change points detected here are points where the behavior of the feature traces deviates significantly from the period since the last detected segment. The core mechanism to detect the boundary points is the GLR test described in the previous subsection. A change point is detected when the GLR distance between the test and learning windows exceeds a fixed threshold δ. For a feature time series, let the most recently detected segment boundary by the segmentation algorithm be at an arbitrary time index $t = r$; then without loss of generality we can define $r = 1$. The decision process needed to detect a new boundary at an arbitrary time index $s > L$ (L is the minimum segment length, $L > 1$), is then performed for all indices $s > L$ by establishing a test window $S_t = \langle x(s), \cdots, x(s+L-1) \rangle$ and a learning window $L_t = \langle x(1), \cdots, x(s-1) \rangle$ and applying a GLR test to the sequences defined by these windows. A new segment boundary is detected whenever the GLR distance for a potential boundary position s, i.e., the GLR distance between the windows, $\langle x(1), \cdots, x(s) \rangle$ and $\langle x(s+1), \cdots, x(s+L-1) \rangle$, denoted by $d(s, s+L-1)$, exceeds the threshold δ. At this point, the time index $s + L - 1$ is called the "detection time" t_D.

The position of the detection time is the indicator that somewhere within the range from the beginning of the current test window till the position t_D a change in the behavior of the traces has been detected. The second part of our algorithm, explained below, determines the optimal position for the boundary within the current test window range $(t_D - L + 1, \cdots, t_D)$, thus, detecting the exact position of the change.

4.1.3 Boundary Position Optimization

The main purpose of this step in the algorithm is to detect accurately the point in the traces where the change occurs. Initially the optimal boundary is assumed to be: $t_D - L + 1$. Then, for all other potential boundary positions within $(t_D - L + 2, \cdots, t_D)$ the GLR distance between the growing learning (W_{GL}) and fixed test (W_{FT}) window is compared with the GLR distance between the fixed learning (W_{FL}) and growing test (W_{GT}) window. The initial sizes of these windows are (ref. Figure 5):

$$\begin{aligned} W_{GL} &: \langle x(1), \cdots, x(t_D - L + 2) \rangle, \\ W_{FT} &: \langle x(t_D - L + 3), \cdots, x(t_D + 1), \\ W_{FL} &: \langle x(1), \cdots, x(t_D - L + 1) \rangle, \\ W_{GT} &: \langle x(t_D - L + 2), \cdots, x(t_D + 1) \rangle \end{aligned}$$

The growing window increases and the fixed test window moves ahead by one at each iteration. The total length composed of both windows is identical in both cases and grows continuously. The learning window size grows from $t_D - L + 2$ to t_D and the test window size from $L-1$ to $2L-1$ at the end of the last iteration. At each iteration the GLR distance between W_{GL} and W_{FT} and the GLR distance between W_{FL} and W_{GT} is calculated. Then the new boundary position is determined based on a second tier comparison of the GLR distances between these two pairs of windows as per:

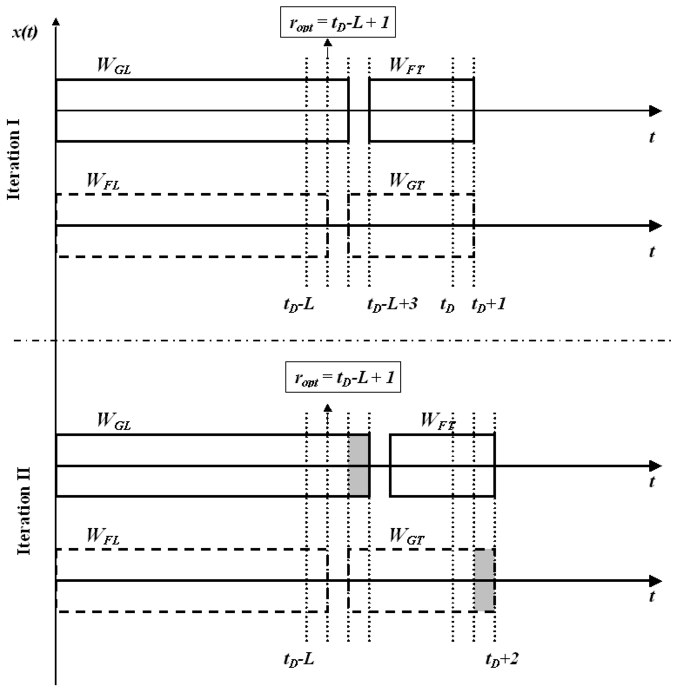


Fig. 5. The first 2 iterations of the boundary position optimization algorithm. The shaded regions of W_{GL} and W_{GT} indicate their growth in the next iteration. The second iteration shown is under the assumption that the GLR distance $d(W_{GT}, W_{FL})$ is more than $d(W_{GL}, W_{FT})$ and the boundary position is not updated.

- If the GLR distance between W_{GL} and W_{FT} is greater than that between W_{FL} and W_{GT}, the current optimal boundary position is updated to the end of the current learning window.
- If on the other hand, the GLR distance between W_{FL} and W_{GT} is greater, then the current optimal boundary position is unchanged as it indicates that adding a new sample to the test window has not affected the statistical similarity of the two segments.

When the last potential boundary position is reached, the algorithm stops and the last allocated boundary position, r, is the optimized boundary. The boundary positions marked after the optimization process are called *alarms*. Algorithm 1 shows the steps for change point detection and boundary optimization.

At the end of the last iteration the learning window size grows from $t_D - L + 2$ to t_D. At one extreme, the position of the optimum boundary is updated at every iteration and the last iteration optimizes the boundary point. On the other extreme, there is no change in the boundary position, i.e., the boundary detected in the first part of the algorithm itself is accurate enough, and the test window size grows from L to $2L - 1$ at the end of the last iteration. For a general case, the final optimal boundary position can be anywhere between the two extreme values $(t_D - L + 1, t_D)$. Thus, the delay between the final boundary position and the detection time of the initial change point is dependent on the minimum window size L. Though the boundary position optimization step introduces a delay in the detection, it is important for minimizing the number of false alarms. This is further explained in the next subsection which describes the clustering of the boundary points.

Algorithm 2 Processes for online alarm clustering.

```
$n$  number of traces;
$a_i$  alarm for trace $i$, $i = 1, 2, ...., n$;
$t(a_i)$  time at which alarm $a_i$ occurs;
$\tau$  maximum time between alarms in a cluster;
$N(a_i)$  alarm cluster in neighborhood of $a_i$;
$A$  Final alarm indicating instability event;

FOR EVERY ALARM $a_i$
Set a timer for $\tau$   timer to inactivate $a_i$ after $\tau$;
Set $N(a_i) = a_i$   add $a_i$ to its own neighborhood;
if ($a_j$ still active and $i \neq j$) then
  Include $a_i$ in $N(a_j)$   add $a_i$ to the neighborhood
                              of other active alarms;
end if

SUBROUTINE FOR TIMER EXPIRATION
if ($N(a_i)$) then
  Delete $N(a_i)$   delete alarm neighborhood;
  Delete $a_i$ from all $N(a_i)$   delete $a_i$ from all other
                                   neighborhoods;
end if
Delete $a_i$

BACKGROUND PROCESS
if ($|N(a_i)| \geq \frac{n}{2}$ ) then
  $A = TRUE$   majority alarm cluster detected;
  Delete $N(a_i)$
  Delete $a_i$ from all $N(a_j)$
  Delete $a_i$
end if
```

4.2 Alarm Correlation

The change detection and boundary position optimization processes of Algorithm 1 are applied to each feature trace and the change points detected are termed *per-feature-trace* alarms. The temporal correlations between the per-feature-trace alarms are used in this section to reduce false alarms and make the detection process more robust against volatility of feature traces.

The clustering of the per-feature-trace alarms is done by using the time difference between them as a distance measure. Then a majority voting rule that requires the largest cluster to have alarms from more than half the number of feature traces is used to generate the final alarm. This two step process to combine the per-feature-traces alarms is defined in Algorithm 2 and is now described in detail.

Step I: In the first step, we consider the feature traces for different values of AS path lengths and AS path edit distances, with each group considered separately, and cluster them in time. We define $N(a_i)$ as the neighborhood of any alarm generated by the i^{th} trace. Also associated with each alarm a_i is a timer that is initialized when the alarm is generated and expires at the end of a threshold τ. The routines for every alarm a_i and those to be run at the expiration of the timer are as given in the pseudocode. Every newly generated alarm for trace i is included in the neighborhood of an alarm $a_j, i \neq j$, as long as the timer for alarm a_j has not expired. A process running in the background keeps track of all the alarm neighborhoods and as soon as any neighborhood contains at least $\lceil n/2 \rceil$ change points, where n is the total number of traces, a first level alarm is generated for the AS path length (or AS path edit distance).

Step II: Next, we cluster the alarms generated by AS path length and AS path edit distance at end of step I and the change points detected by the withdrawal traces in time. If the cluster strength is 2 or more elements, an instability-alarm is generated. The different pairs of features are preserved in the combination scheme for possible classification of different kinds of instabilities.

4.3 Preventing False Alarms

The suppression of false alarms is achieved through the use of a diverse feature set as well as through the various steps in our detection process. The use of AS path length and path edit distance features helps lower the false alarm rate and minimize the detection delay by maintaining a low clustering threshold τ. This is a significant advantage over using just volume-based (announce and withdrawal message volume) detection.

In order to ensure that we detect the instabilities correctly and do not miss any, the individual feature trace alarms are required to be as precise as possible. This is ensured by the boundary position optimization step in the segmentation process. In the absence of the optimization step, valid instability indicator alarms may be missed when the initial step for change detection gives a delayed change point that does not fall within the clustering distance of the change points of the other traces. The optimization step gives an alarm at the exact segment boundary, facilitating its clustering with alarms from other traces and thereby generating a valid instability indicator alarm. While the cluster threshold τ may also be increased in order to include delayed change points, this can lead to an increased number of false alarms and delay the detection of the instability.

4.4 Complexity of the Detection Algorithm

The complexity of the proposed scheme can be considered in parts and independently for every peer. The first step of the algorithm is feature extraction and the most expensive in that is AS path edit distance calculation that is performed for every update message received per peer. This is typically $O(n^2)$ complexity where n is the number of ASes in the AS path. Since the normal value of n is around 5 and the worst case value is around 20, this will not prove to be very expensive. The next step is the 'Segment Boundary Detection' (Section 4.1.2) which involves the calculation of covariance matrices and pooled covariance matrices. These operations can however be performed incrementally for every new data point received [3] and thus the GLR calculation is just $O(\rho^2)$ where ρ is the AR order that decides the order of these matrices. For our experiments we chose $\rho = 1$ and typically the value of ρ is small enough for these operations to be quite inexpensive.

The next step of 'Boundary Position Optimization' (Section 4.1.3) calculates the GLR distance between different combinations of the learning and test window sizes. However, again for the growing learning, fixed learning and the growing test windows this calculation can be done in an incremental manner. The only expensive calculation is the covariance estimation for the fixed test window which is $O(L^2)$. It is important to note here that this expensive step is performed only after an initial segment boundary has been detected. The other steps of the algorithm such as 'Median filtering', 'Alarm correlation' etc. are all of constant order and do not cause any additional complexity. So the overall complexity of the algorithm is just dependent on ρ and L, but is dominated by L since typically $L \gg \rho$.

4.5 System Evaluation

This section discusses the system requirements for implementing our methodology in a real-time environment, focusing specifically on memory usage and execution time. We implemented our system on a simple Notebook PC with a 2.16GHz Intel processor and 2.00GB of RAM. Since the online implementation of our detection scheme was not operationally possible we implemented it offline for our experiments using Matlab. For the longest duration of the data that we processed (i.e. 11 days for the January 2006 event), the memory usage for the process did not exceed 121Mb and the execution time was 9.813 seconds. This included the resources needed for covariance calculations over the entire dataset, which in an online implementation will be done in an incremental step for each time sample of 5 minutes. The only other expense that our detection scheme will incur is the storage of the time series data and that requirement depends on the number of peers used, the duration of data stored etc.

5 Root Cause Location Analysis

This section presents the details of the Alarm Classifier component of the proposed system architecture shown in Figure 1. The objective of this component of the system is to pinpoint the exact location where the root cause of the instability event may have occurred before the instability propagated to the local router. We focus on just identifying the AS where the root cause event occurred and call it the *root-cause-AS*. While other information like the time when the root cause event happened, the actual network prefixes that it affected, etc. could also be obtained, we focus on the root-cause-AS since it is the most useful in terms of controlling the spread of the instability.

In order to locate the root-cause-AS, we first introduce the concept of *first-changed-AS*. This is the AS (in the update messages) where the first change is seen between AS paths received for routes to the same destination in consecutive update messages. The method that we employ to find the first-changed-AS is an extension of the AS path edit distance feature extraction process. We first use the same algorithm that is used for edit distance calculation to align the successively received AS paths, say $ASPath_1$ and $ASPath_2$. Let $l_1 = |ASPath_1|$ and $l_2 = |ASPath_2|$ be their respective lengths. Also, let the aligned AS paths be referred to as $ASPath'_1$ and $ASPath'_2$ respectively and their common length after alignment be l. We denote by $ASPath_i(k)$ the $k-$th AS in $ASpath_i$. The first-changed-AS, $ASPath(l - i^* + 1)'$, then corresponds to i^* given by

$$i^* = \arg\min_i \left[ASPath(l-i)'_1 \neq ASPath(l-i)'_2\right], \forall i = \{0, \cdots, l\}$$

where $l = \max(l_1, l_2)$. Thus we trace the AS paths in the reverse direction i.e. starting from the origin AS and identify the AS where the first change in the AS path is observed. To illustrate this better, consider the following pair of AS paths:

6762_701_6453_17557_9557
6762_17557_9557

After alignment they can be represented as:

6762_701_6453_17557_9557
$6762\_ - \_ - \_17557\_9557$

Here the dashes represent the deletion of 2 ASes in the first AS path to obtain the second. Using the above alignment we can identify AS17557 as the first-changed-AS.

The basic data-set used for this analysis is update messages exchanged by the BGP routers. In order to ensure the absence of transient route fluctuations, we first apply the route-flap damping process of the BGP protocol [15] to suppress redundant routes that flap persistently. Our experiments used the default values of the route flap damping algorithm parameters according to those set on Cisco routers [18]. The root cause analysis is then applied on the unsuppressed routes that are not flapping. After obtaining the unsuppressed non-flapping route changes, the next step is to obtain the first-changed-AS for each route change. Since at any given time there are a large number of route changes, a large set of first-changed-ASes is obtained. We term this as the set of *candidate-root-cause* ASes. The mode of the frequency distribution of the candidate AS number(s) is then marked as the actual root-cause-AS (ASes) where the event occurred. It is important to note that this method is only determines the AS where the root cause originated and not the actual root cause of the event.

Any set of candidate-root-cause ASes will include ASes where path changes due to transient route changes, short-lived failure events, policy shifts, session resets as well as the large instability events are seen. By considering only the path changes for a given period after the instability has been detected and eliminating persistently flapping routes, we maximize the likelihood of the set of candidate-root-cause ASes being formed only by path changes that are induced by the root cause and related to the instability. But to further account for the high rate of update messages and path changes seen at most core BGP routers, we select the n most frequently occurring candidate-root-cause ASes as the set of actual root-cause-ASes. The value of n is chosen based on the volume of path changes usually seen at the router and on the error requirements of the system. Section 7 discusses the effect of the choice of n on the detection accuracy.

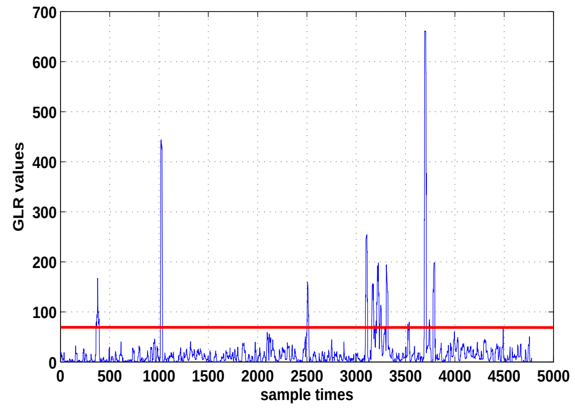


Fig. 6. The GLR values for the withdrawal messages received from the peer in AS6893 for the entire month of December 2002. The straight horizontal line indicates the $95th$ percentile of the data.

6 Parameter Estimation

In this section we describe the methodology for selecting the parameters required to implement our detection algorithm.

Order of the median filter (m): In order to avoid capturing the transient peaks, we median filter the feature traces. It has been shown that after a normal routing change the BGP convergence process can last upto 15 minutes before a new stable route is installed [10]. We consider such spikes in the traces as normal routing changes and suppress them by setting the median filtering order accordingly.

Minimum window size (L): We select this to be at least twice the median filter order, so as to avoid choosing a window that is entirely smoothed out. On the higher side, the limit on the window size is the size of the smallest segment of change that is to be detected. It has been recommended that [1]

$$L \leq 0.7c \tag{12}$$

where c is the size of the smallest segment of change to be detected. Given the timescales of the BGP instabilities, we select L to be 20 samples (100 minutes of data).

AR Order (ρ): The selection of the AR order is done according to the Akaike's Information Criteria (AIC)

Data for	Month	RRC	No. of peers	Peers that sent messages
Moscow blackout	May 2005	rrc05, Vienna	3	AS1853,AS12793, AS13237
SQL/Slammer Worm	January 2003	rrc04, Geneva	3	AS513, AS559, AS6893
Nimda Worm	September 2001	rrc04, Geneva	3	AS513, AS559, AS6893
Code Red II Worm	July 2001	rrc04, Geneva	3	AS513, AS559, AS6893
AS3300,AS2008 AS-path error	October 2001	rrc03, Amsterdam	4	AS3257, AS3333, AS6762, AS9057
AS9121 Routing Table Leak	December 2004	rrc05, Vienna	3	AS13237, AS12793, AS1853
AS3561 Improper Filtering	April 2001	rrc03, Amsterdam	3	AS3257, AS3333, AS286
AS3356 Incorrect import to IGP	October 2005	rrc01, London,	4	AS13237, AS8342, AS5511, AS16034
Panix Domain Hijack	January 2006	RouteViews, Oregon	4	AS12956, AS6762, AS6939, AS3549

TABLE 2
The datasets used to test for different instabilities.

[2]. To estimate the best value of ρ, we use one of the smoothed data sets and segment it into windows of size $L = 20$. We then impose AR orders from 1 to 10 on each of these windows and note the value of the AR order that gives the minimum AIC value for each of the windows. We then select the AR order that gives the minimum AIC value for maximum number of windows of the selected size. Based on this method we estimated $\rho = 1$ to be appropriate for the data.

GLR Threshold (δ): δ affects the sensitivity as well as accuracy of the detection. While small values result in almost all the routing changes being captured, a very large value can lead to missing even the significant instabilities. To select a suitable δ, we use data from a period of normal operation. We run the detection algorithm with window size $L = 20$ and a very high δ. As the threshold is very high, no segment change is detected during this process and we obtain the GLR ratios over all the segments. Then, we set δ to be greater than the $95th$ percentile of these (ref. Figure 6). This process obtains a δ that will not be crossed 95% of the time, so that only the update message surges that are relevant will cause a segment boundary to be detected.

Cluster distance threshold for the alarms (τ): The value of τ is dependent on factors like number of peers, nature of the peers, topology, etc. We empirically set the value of the clustering threshold to be 40 minutes guaranteeing that any instability will be detected within an hour of its detection in the feature traces at the router.

7 Performance Evaluation

In order to evaluate the performance of the proposed system, we tested it on real BGP update messages collected at the Réseaux IP Européens (RIPE) remote route collectors (RRC's) and the Routeviews, Oregon server. We only used data from those route collectors that have a direct BGP connection with peers located at the same exchange points in order to avoid any impact of the collection process itself on the data [24]. We used five-day long traces for testing the detection of different types of instabilities. The raw data collected was in the form of the update message logs received during 5 minute intervals. For each period of instability, we use data from different route collectors. This was done because some instabilities have a significant impact only at specific RRCs and also to test the scheme for data from diverse sources. Table 2 gives a list of the different data traces we used for the different instability types. We observed that even though the RRCs listed in the table peer with many more BGP routers [16], only the peers mentioned in the table sent a statistically significant amount of update messages during the particular periods we tested on.

7.1 Evaluation of the Detection Mechanism

In this section we test our detection mechanism on three types of instability events and present the associated results. The results of the evaluation of our detection algorithm are shown in Table 3. Note that the parameter values estimated in Section 6 were used for all these experiments and no additional tuning of the parameters was necessary.

Worm Attacks: We tested our scheme for detecting the instabilities caused by three worm attacks on the Internet that have occurred in the past and are known to have impacted the BGP routing mechanisms. Figure 7 shows the results of each step of the detection algorithm as applied to the different feature traces for one peer during the five days around the period of the Slammer worm attack (from 23rd to 27th January 2003).[1] Figure 7(a) shows the median filtered traces for the withdrawal volume and AS path length features, in which the anomaly is seen as a sharp peak. Similarly, Figure 7(b) shows the median filtered traces for the AS path edit distance features. In part (c) of the figure we show the results of the change point detection step for each feature trace and the last figure at the bottom shows the final alarm. We also applied the detection mechanism on the periods around the Nimda and Code Red II worm attacks of September and July 2001 respectively. Table 3 shows the duration of these events and the number of alarms raised per peer with our detection algorithm. The number of false alarms generated are also given. The detection of the worm attacks at a majority of the peers indicates that their impact was global.

Equipment Failures: Around the 25th of May 2005, there was a blackout in Moscow that is known to have caused the Moscow Internet Exchange (MSIX) to be shut

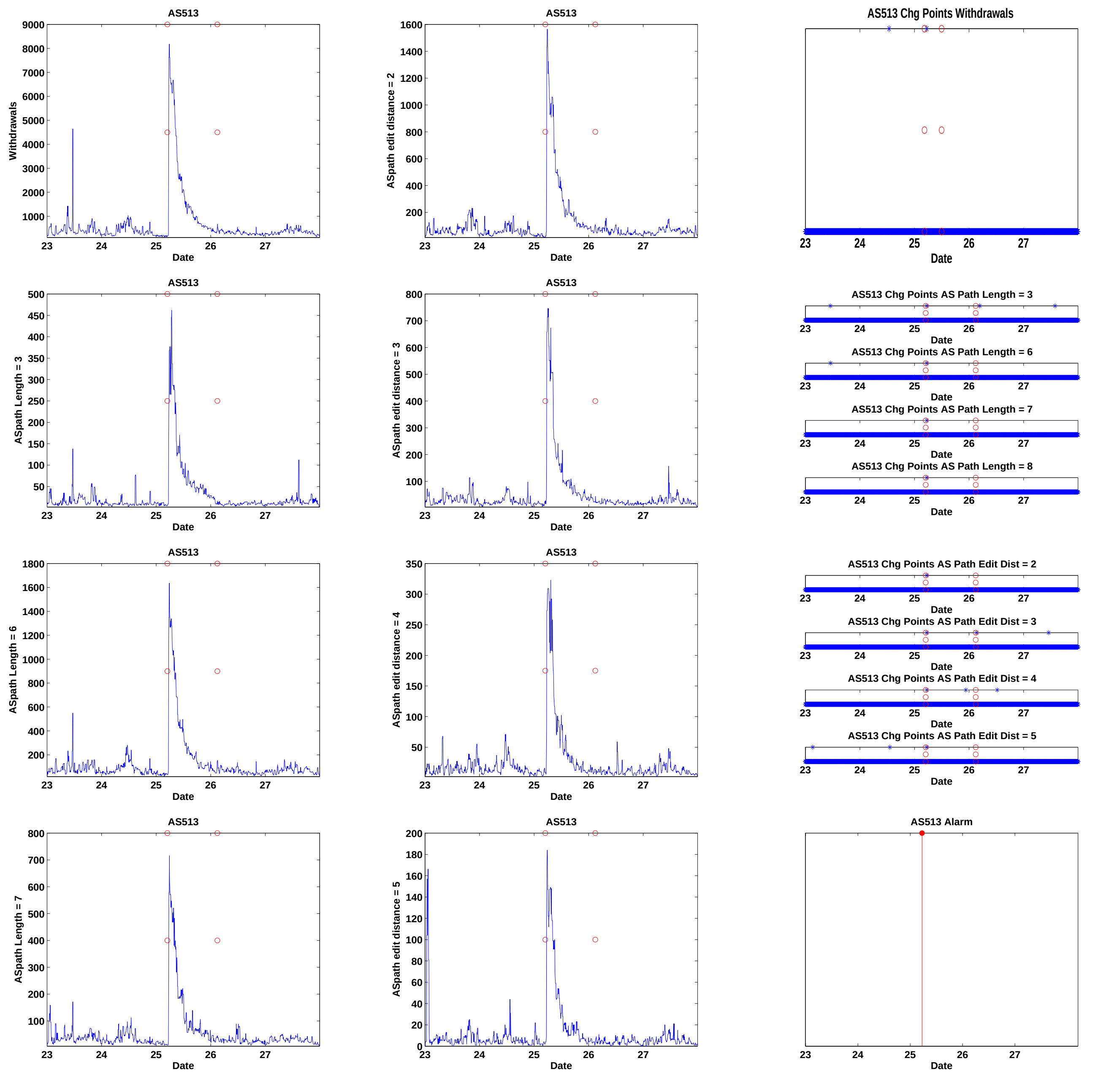
1. Detection results for the other instabilities are similar and omitted due to lack of space.

Event	Slammer Worm			Moscow Blackout			Code Red II Worm		
Month	Jan 2003			May 2005			Jul 2001		
AS Num.	AS513	AS559	AS6893	AS1853	AS12793	AS13237	AS513	AS559	AS6893
Duration*	22hrs	21hrs.	30hrs	6hrs.	8hrs	<2hrs.	11hrs	11hrs	11hrs
Our Detection Algorithm									
# Alarms per event	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
# False Alarms	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
EWMA-mechanism using only volume features									
# Alarms per event	246	254	271	2	0	47	0	20	13
# False Alarms	6	14	6	2	47	0	2	11	4
EWMA-mechanism using all our features									
# Alarms per event	209	134	156	0	1	0	0	14	6
# False Alarms	1	12	4	0	1	0	0	2	0
PCA based mechanism using only volume features									
# Alarms per event	22			1			1		
# False Alarms	14			8			8		
Wavelet based mechanism using only volume features									
# Alarms per event	2	0	1	0	0	0	4	0	1
# False Alarms	0	0	0	2	5	2	0	4	0
Wavelet based mechanism using all our features									
# Alarms per event	0	0	0	0	0	0	4	0	0
# False Alarms	0	0	0	1	3	0	0	4	0

Event	Nimda Worm			AS9121 Leak			AS3356, error in import to IGP.			
Month	Sep 2001			Dec 2004			Oct 2005			
AS Num.	AS513	AS559	AS6893	AS1853	AS12793	AS13237	AS13237	AS8342	AS5511	AS16034
Duration*	>48hrs	>48hrs	>48hrs	5hrs	8hrs.	6hrs	3hrs.	5hrs	9hrs.	9hrs
Our Detection Algorithm										
# Alarms per event	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**1**
# False Alarms	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**1**
EWMA-mechanism using only volume features										
# Alarms per event	77	20	236	60	14	13	31	0	9	34
# False Alarms	3	10	3	12	1	6	12	13	7	14
EWMA-mechanism using all our features										
# Alarms per event	0	0	153	23	11	13	25	0	9	21
# False Alarms	1	3	1	7	0	1	6	10	2	1
PCA based mechanism using only volume features										
# Alarms per event	2			14			5			
# False Alarms	0			3			4			
Wavelet based mechanism using only volume features										
# Alarms per event	9	0	8	1	5	1	0	3	0	0
# False Alarms	8	0	0	0	0	0	0	1	3	0
Wavelet based mechanism using all our features										
# Alarms per event	8	1	15	4	7	3	0	8	1	0
# False Alarms	6	0	0	1	0	0	0	1	1	0

Event	AS3561 Filtering error			AS3300,AS2008 AS-path error			Panix.com Domain Hijack			
Month	April 2001			Oct 2001			Jan 2006			
AS Num.	AS3257	AS3333	AS286	AS3257	AS6762	AS3333	AS12956	AS3549	AS6762	AS6939
Duration*	8hrs	2hrs	16hrs	5hrs	7hrs	6.5hrs	10hrs	<1hr.	22hrs.	20hrs
Our Detection Algorithm										
# Alarms per event	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**
# False Alarms	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**3**	**3**	**0**
EWMA-mechanism using only volume features										
# Alarms per event	6	1	31	4	28	13	20	7	0	5
# False Alarms.	5	4	21	8	2	8	6	1	0	7
EWMA-mechanism using all our features										
# Alarms per event	0	0	0	0	68	14	2	0	0	2
# False Alarms	0	0	15	1	1	3	3	0	0	1
PCA based mechanism using only volume features										
# Alarms per event	5			2			32			
# False Alarms	4			16			48			
Wavelet based mechanism using only volume features										
# Alarms per event	2	0	2	0	1	4	4	0	0	0
# False Alarms	4	2	9	1	5	2	1	4	2	6
Wavelet based mechanism using all our features										
# Alarms per event	1	0	0	0	0	0	3	0	0	0
# False Alarms	6	1	5	0	4	2	0	0	1	4

TABLE 3
The results for our detection algorithm and schemes proposed in literature for various events. The entries showing zero alarms in the true alarm clusters represent the missed alarms. *Duration of an event is the approximate amount of time for which the surge in the volume of update messages lasts.



(a) Median filtered feature traces (b) Median filtered feature traces (c) Optimized Change Points per trace and Alarm

Fig. 7. Results of the different steps of the algorithm for the Slammer worm of January 2003. The data is shown for 23rd to 27th of January; Parameters: $L = 100$mins, $\rho = 1$, $\delta = 95^{th}$ quantile of the normal period GLR, $\tau = 50$mins data points. The circular markers indicate the period of the Slammer worm. In Figure(c) the markers at the upper limit of the y-axis show the alarms after boundary position optimization. The final plot in (c) shows the alarm obtained after correlating the alarms from these traces.

down for several hours [13]. Though this did not affect the Internet on a global scale, some of the ISPs peering at this exchange did lose connectivity for an extended period, leading to routing instabilities visible to some parts of the Internet. We observed that the number of announcement and withdrawal messages received at the RIPE rrc05 in Vienna surged during this period and our algorithm raised an alarm for the peer in AS12793 for this instability. Table 3 shows the detection results for this event for this dataset.

BGP Misconfiguration and Hijack Events: Our scheme was also tested for the detection of a number of instabilities due to BGP misconfigurations. Five different instability events were considered. These misconfiguration events had causes as diverse as: incorrect import to IGP, domain hijack, improper filtering and private AS number leak. As shown in the results in Table 3, the correct detection of the events was seen at a large number of peers. Peers where an alarm is not generated in Table 3 correspond to those where the impact of the event was felt only for a short duration.

7.2 Comparison with Other Detection Mechanisms

In this section, we compare the proposed detection scheme with three existing mechanisms: an adaptive EWMA based scheme [17], a PCA based scheme [9] and a wavelet based scheme [28]. The EWMA based mechanism can accommodate any linear trends or baseline shifts in the feature time series and uses the number of best routes seen exiting a specific PoP by the local route collector as the feature for detection. This feature is very close to counting the number of updates sent by a particular router, i.e. volume features used in our algorithm. The other two mechanisms work specifically on volume traces.

The detection results for the various schemes are presented in Table 3. For the 9 datasets covering 9 events, the number of false alarms generated is 235 for the EWMA based scheme, 103 for the PCA based scheme and 61 for the wavelet based scheme. In contrast, our algorithm generates only 16 false alarms. Due to the inherent nature of the PCA based scheme, the alarms are obtained by finding the principal component based residuals of the matrix constructed from the volume traces from each peer as a column. Thus the results are presented "per event" rather than "per peer" as in the other schemes.

In order to reduce the false alarms in the existing schemes, we also implemented them using all our features. The final alarm is thus generated only after the alarms from all the feature traces coincide. Using all the features reduces the overall number of false alarms in the EWMA scheme to 76 and in the wavelet based scheme to 40. However, with more constrained alarm generation, the number of missed alarms also increases for the EWMA and wavelet based scheme for most cases. For the PCA based scheme, using the edit distance and path length features does not produce statistically significant results, since we are already looking at only those AS path length and edit distance values for which the number of messages under normal circumstances is quite low. Our detection algorithm performs significantly better in terms of the false alarm rate and slightly better in terms of the missed alarms (with longer learning periods we expect much fewer missed alarms).

7.3 Evaluation of the Root Cause Location Scheme

We applied the method discussed in Section 5 for identifying the root-cause-AS on the data from the duration of different events. The results are presented (in Table 4) for six instability events: four of these are due to misconfigurations, one is due to the equipment failure caused by power outage in Moscow in May 2005 and one due to the hijacking of the domain `panix.com` in January 2006. We do not present any results for the instabilities caused by worm attacks. This is primarily due to unavailability of any information regarding the AS where the event started. Also, since the method of the worm attack propagation is random, its effect on the routing process cannot be traced. Hence, there was no means of validating the results of our root-cause-AS identification technique applied on data from the worm attack events.

From Table 4 we can see that our methodology was able to locate the root-cause-AS for most of the events and datasets irrespective of the actual root cause. For the BGP misconfiguration error events in December 2004, October 2005, April 2001, and October 2001 a single root cause AS was known and it was successfully identified by our technique. However, for the events in May 2005 and January 2006 a single root cause is not known and we discuss the results for these two cases in detail.

On 25th May 2005, a power failure in Moscow affected the MSIX, causing loss of connectivity between several ASes peering there. Hence, a single AS cannot be identified as the originating AS for this event. It is known that a large number of networks in the Asia-Pacific region are reached through peering at MSIX and as a result many of the ASes whose connectivity was affected due to the event were from the APNIC blocks [13]. Also, many Russian ASes were affected by the power failure [13]. Thus, we deem any AS that falls in either of these categories as a possible root-cause-AS for this event. We can see from the results that our techniques were able to capture many such ASes as the top candidate root-cause-ASes for data from the period of this event. A list of some of these ASes that appear in our results is as follows: AS7473: Singtel (APNIC); AS8342: Russia Telecom, RUNet; AS20485: JSC Company TransTelecom (Participant at MSIX); AS9198: KazakhTelecom; AS7693: COMNET-TH (APNIC); AS4795: INDOSAT (APNIC); AS7610: SINGAREN-AS-AP (APNIC); AS5568: RBNet Russian Backbone; AS9270: APAN-KR-AS (APNIC).

On January 22, 2006, the domain `panix.com` was hijacked, leading to BGP instabilities [14]. The event was caused because AS27506 advertised routes to several prefixes that it did not own. Thus, the known root-cause-AS for this event is AS27506. However, the major Internet Service Providers (ISPs) downstream from AS27506 did not have correct filters in place and advertised these incorrect routes which in turn led to the routing message storm and caused a major instability. Thus even though the originating AS for these incorrect routes was AS27506, the ASes that actually caused a long-term instability were the ISPs such as UUNet (AS701, AS702), Time Warner (AS4323) etc. and are thus the actual root cause ASes for the event. Our scheme thus correctly identified the major ISPs as the root-cause-ASes for this event.

In Table 4, we used the top 5 candidate root-cause-ASes (i.e. $n = 5$) as identified by our algorithm. By providing top 5 candidates we reduced the total number of candidates by 99.6%. Overall, the number of missed *root cause ASes* were 3 in the 20 datasets that we considered. If we use only the top *candidate root cause AS*, we miss

Event and root cause AS	Peer AS	Top 5 Candidate Root Cause ASes	No. of Candidate Root Cause ASes
Apr 2001, AS15412	AS3257	**AS15412**, AS8708, AS5727, AS855, AS3749	1192
Apr 2001, AS15412	AS3333	**AS15412**, AS9057, AS701, AS1239, AS209	1080
Apr 2001, AS15412	AS286	-***	-***
Dec 2004, AS9121	AS12793	**AS9121**, AS701, AS14359, AS6762, AS702	5373
Dec 2004, AS9121	AS13237	**AS9121**, AS701, AS6762, AS7018, AS1239	8559
Dec 2004, AS9121	AS1853	**AS9121**,AS5588, AS702, AS721, AS6762	4863
Oct 2005, AS3356	AS5511	AS6389, **AS3356**, AS702, AS6167, AS3464	466
Oct 2005, AS3356	AS16034	**AS3356**, AS8210, AS701, AS2907, AS18566	725
Oct 2005, AS3356	AS13237	**AS3356,** AS8342, AS6389, AS852, AS3216	1080
Oct 2005, AS3356	AS8342	AS3662, **AS3356,** AS6453, AS18662, AS9121	257
Jan 2006, -*	AS12956	AS17676, AS22822, **AS4323,** AS9837, AS6389	1734
Jan 2006, -*	AS6762	AS2529, AS5417, AS8992, AS8342, AS7738	285
Jan 2006, -*	AS6939	AS5511, **AS701,** AS22351, AS1299, AS7303	113
Jan 2006, -*	AS3549	AS4766, **AS702,** AS4621, AS9121, AS18747	340
Oct 2001, AS3300	AS3257	AS3356, AS12083, AS12302, AS10242, AS11042	270
Oct 2001, AS3300	AS3333	AS701, AS3356, AS12083, AS1239, AS3561	181
Oct 2001, AS3300	AS6762	AS701, AS1239, **AS3300**, AS3356, AS7018	262
May 2005, -**	AS12793	AS3356, **AS7473, AS8342,** AS1273, **AS9198**	629
May 2005, -**	AS13237	AS3356, **AS7473, AS8342,** AS1299, AS7018	706
May 2005, -**	AS1853	AS1273, **AS20485, AS7693, AS7473,** AS6667	526

TABLE 4
Root cause location analysis results for different instability events conducted over a period of 1 hour after the event detection flag time. *Even though for the January 2006 event the root cause is identified as AS27506, very few messages are seen from this particular AS. Instead, messages for a number of affected prefixes show candidate root-cause-ASes as some big ISP ASes like UUNet (AS701), TimeWarner (AS4323) etc. **The root-cause-AS for this MSIX failure event is in fact a large group of ASes that were affected. ***In case of the peer in AS286 for the April 2001 event duration the highest frequency of messages was flapping routes and duplicate announcements, as a result a clear candidate root-cause-AS was not identified.

13 of the 20 root cause ASes. Instead if we use the top 3, we miss 4 of the 20 datasets and achieve an average reduction of 99.8% in the total number of candidates. Thus, based on our error requirements we can select a value of n and use the top n *candidate root cause ASes* to identify the correct *root cause AS*.

7.4 Parameter Sensitivity

In this section we discuss the sensitivity of our algorithm to the estimated parameters. For each of the parameters discussed in Section 6, we varied the possible values and observed the impact on the number of false and missed alarms generated by our algorithm. The analysis was done using the data for the BGP misconfiguration events described above. The total number of per peer alarms that can be raised for this dataset is 17, whereas our algorithm generates 14.

It is important to note here that the parameter sensitivity is studied only in terms of the number of false/missed alarms and not the detection delay. This is because even though all the parameters will influence the detection delay to some extent, the clustering threshold (τ) dominates the detection delay caused. Thus the lower the clustering threshold, the lower will be the final detection delay and we select the optimum value of τ based on the number of false/missed alarms.

- Median Filter Order (m): In Figure 8 we plot the incidence of false and missed alarms as a function of the increasing values of m. As can be seen, $m = 7$ leads to the lowest number of false as well as missed alarms. Thus, we select the median filter order of

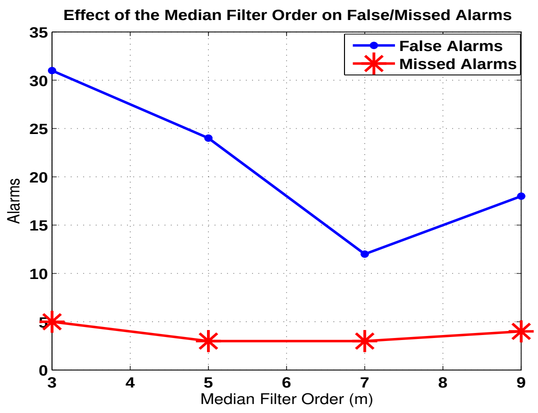


Fig. 8. Incidence of missed and false alarms as a function of the median filter order m.

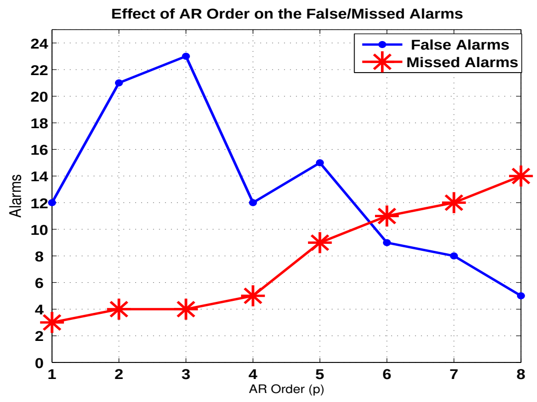


Fig. 9. Incidence of missed and false alarms as a function of AR order ρ.

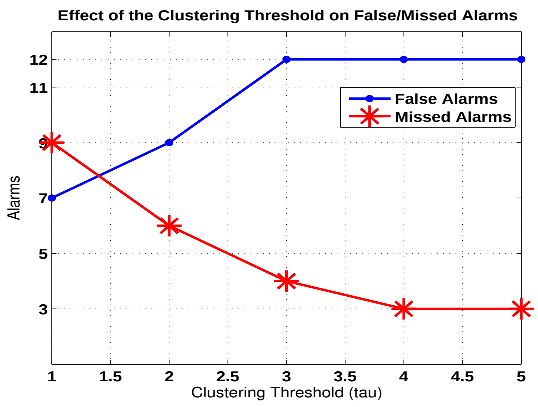


Fig. 11. Incidence of missed and false alarms as a function of the clustering threshold τ.

7 which is equivalent to $15minutes$ for our results. The value of the initial learning window size (L) is dependent on m and is selected to be just greater than $2m$.

- AR Order (ρ): The impact of the AR order on the number of false and missed alarms is shown in Figure 9. As can be seen, $\rho = 1$ leads to the optimum performance in terms of both missed and false alarms.
- GLR Threshold (δ): Figures 10 (a) and (b) depict the incidence of false and missed alarms as a function of increasing values of δ. As can be seen, the lower values of δ are indiscriminate and generate alarms for all possible deviations from normal behavior. Thus, we select $\delta = 0.95$ in order to obtain the optimum number of false and missed alarms.
- Clustering threshold (τ): As can be seen in Figure 11, $\tau = 4$ leads to the minimum number of both missed and false alarms. Thus we set $\tau = 4$, which sets the worst case detection delay at 2τ i.e. 40 minutes (as the alarm correlation is a 2-step process) for the five minute sampling interval.

8 Conclusion

This paper presents a mechanism for capturing the statistical changes in features extracted from BGP update message data for online detection of routing instabilities on a single router. The features like AS path length and AS path edit distance that reflect the actual topological changes rather than just volume anomalies were found to be very useful in characterizing the behavior of the topology under stress. Also, the spatial and temporal correlations amongst these features were used effectively to minimize the number of false alarms. The originating ASes for the root cause of the instabilities were also accurately identified based on AS path changes. We have shown the validity of our approach through extensive evaluations with different types of instabilities. We believe that our mechanism can complement most of the existing work on improving BGP through changes to the message handling procedures.

References

[1] U. Appel, A. Brandt, "Adaptive Sequential Segmentation of Piecewise Stationary Time Series," *Information Sciences* vol. 29, pp. 27-56.

[2] H. Akaike, " Information theory as an extension of the maximum likelihood principle" *Proc. of ISIT,* pp. 267-281, 1973.

[3] T. Chan, G. Golub, R. LeVeque, "Algorithms for Computing the Sample Variance: Analysis and Recommendations," *The American Statistician,* Vol. 37, No. 3, pp. 242-247, Aug. 1983.

[4] J. Cowie, A. Ogleski, B. Premore, and Y. Yuan, "Global routing instabilities during Code Red II and Nimda worm propagation," Technical Report, Renesys Corporation, December 2001.

[5] S. Deshpande, M. Thottan and B. Sikdar, "Early Detection of BGP Instabilities Resulting from Internet Worm Attacks," Proc. of IEEE GLOBECOM, pp. 2266-2270, Dallas, TX, November 2004

[6] A. Feldmann, O. Maennel, Z. Mao, A. Berger and B. Maggs, "Locating internet routing instabilities," *ACM SIGCOMM Computer Communication Review,* vol. 34, no. 4, pp. 205-218, August 2004.

[7] T. K. Ho, "Mirage: Interactive Tools for Pattern Discovery," *Proc. of ICPR,* Cambridge, U.K., August 22-26, 2004, pp. 509-512.

[8] X. Hu and M. Mao, "Accurate Real-time Identification of IP Prefix Hijacking," *Proceedings of IEEE Symposium on Security and Privacy,* pp. 3-17, Oakland, CA, May 2007.

[9] Y. Huang, N. Feamster, A. Lakhina and J. Xu, "Diagnosing network disruptions with network-wide analysis," *Proc. of ACM SIGMETRICS,* pp. 61-72, San Diego, CA, June 2007.

[10] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. "Delayed Internet Routing Convergence," *Proc. of ACM SIGCOMM,* Stockholm, Sweden, August 2000.

[11] M. Lad, X. Zhao, B. Zhang , D. Masey and L. Zhang, " Analysis of BGP Update Surge during the Slammer Worm Attack," *5th International Workshop on Distributed Computing (IWDC),* 2003.

[12] M. Lad, D. Massey, D. Pei, W. Wu, B. Zhang and L. Zhang, "PHAS: A Prefix Hijack Alert System," *Proc. of USENIX Security Symp.,* 2006.

[13] North American Network Operators Group mailing list, `http://www.merit.edu/mail.archives/nanog/`

[14] North American Network Operators Group mailing list, "oof. panix sidelined by incompetence... again," `http://www.merit.edu/mail.archives/nanog/2006-01/ msg00483.html`

[15] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," RFC 1771, IETF, March 1995.

[16] Réseaux IP Européens Network Coordination Center, `http://www. ripe.net/projects/ris/rawdata.html`.

[17] M. Roughan, T. Griffin, M. Mao, A. Greenberg, B. Freeman, "Combining Routing and Traffic Data for Detection of IP Forwarding Anomalies," *Proc. of ACM SIGCOMM NeTs Workshop,* Portland, OR, Aug, 2004.

[18] "The SSFNet Project, " `http://www.ssfnet.org`

[19] S. Teoh, S. Ranjan, A. Nucci and C.-. Chuah, "BGP eye: a new visualization tool for real-time detection and analysis of BGP anomalies," *Proceedings of the International Workshop on Visualization for Computer Security,* Alexandria, VA, November 2006.

[20] S. Teoh, K. Zhang, S. Tseng, K. Ma and F. Wu, "Combining Visual and Automated Data Mining for Near-Real-Time Anomaly Detection and Analysis in BGP," *In Proc. of ACM VizSEC/DMSEC,* October 2004.

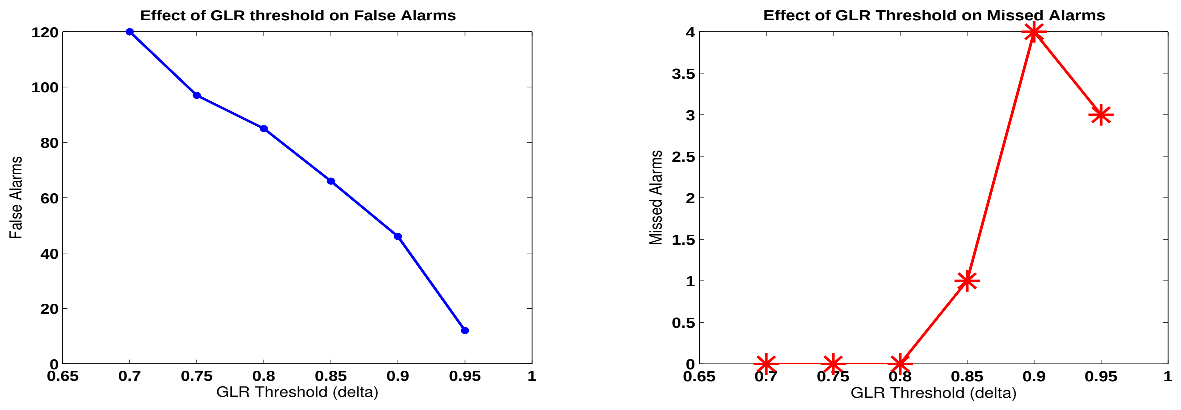
[21] R. Wagner and M. Fisher, "The string to string correction problem," *Journal of Assoc. Comp. Mach.,* 21(1):168– 173, 1974.

[22] H. Wang K. Chang, D. Chiu, C. Lui "Characterizing the Performance and Stability Issues of the AS Path Prepending Method: Taxonomy, Measurement Study and Analysis" *Proc. of ACM SIGCOMM Workshop,* April 2005.

[23] J. Wang, X. Chen and W. Gao, "Online selecting discriminative tracking features using particle filter," *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1037-1042, San Diego, CA, June 2005.

[24] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. F.Wu, and L. Zhang, "Observation and Analysis of BGP Behavior Under Stress," *Proc. of ACM IMW,* Marseille, France, Nov 2002.

[25] T. Wong, V. Jacobson and C. Alaettinoglu, "Internet routing anomaly detection and visualization," *Proceedings of IEEE DSN,* pp. 172-181, 2005.



(a) False alarms

(b) Missed alarms

Fig. 10. Incidence of missed and false alarms as a function of the GLR threshold δ.

[26] J. Wu, Z. Mao, J. Rexford and J. Wang, "Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network," *Proceedings of NSDI,* Boston, MA, May 2005.

[27] K. Xu, J. Chandrashekhar and Z. Zhang, "A first step towards understanding inter-domain routing dynamics," *Proceedings of ACM SIGCOMM MINENET,* Philadelphia, PA, August 2005.

[28] J. Zhang, J. Rexford and J. Feigenbaum, " Learning-Based Anomaly Detection in BGP Updates," *Proc. of ACM MineNet Workshop,* 2005.

[29] Z. Zhang, Y. Zhang, Y. Hu. Z. Mao and R. Bush, "iSPY: Detecting IP Prefix Hijacking on my Own," *Proc. of ACM SIGCOMM,* Seattle, WA, August 2008.

[30] C. Zheng, L. Ji, D. Pei, J. Wang and P. Francis, "A light-weight distributed scheme for detecting IP prefix hijacks in real-time," *Computer Communication Review,* vol. 37, no. 4, pp. 277-288, October 2007.

PLACE PHOTO HERE

Shivani Deshpande Shivani Deshpande received the B.E degree in Electronics and Tele-Communication Engineering from Maharashtra Institute of Technology, Pune, India, the M. S. degree in Electrical Engineering and Ph.D in Electrical Engineering from Rensselaer Polytechnic Institute, Troy, NY, USA in 2000, 2003 and 2007, respectively. She is currently working on improving traffic classification techniques as a Software Engineer at BlueCoat Systems Inc., Sunnyvale, CA. Her research interests are traffic characterization, machine learning, routing protocols and network security.

PLACE PHOTO HERE

Marina Thottan Marina Thottan is a Member of Technical Staff in the Center for Networking Research at Bell Laboratories, Alcatel-Lucent. She holds a Ph.D. in Electrical and Computer systems engineering from Rensselaer Polytechnic Institute in Troy, NY. Dr. Thottan is active in the fields of wire line networking and network management and has served as a program committee member for several conferences in these areas. Her research publications have appeared in a number of ACM, and IEEE conferences and journals. Her current research interests are in the areas of novel network and switch architectures and high speed optical networks. She is a member of the IEEE and the ACM.

PLACE PHOTO HERE

Tin Kam Ho Tin Kam Ho leads the Statistics and Learning Research Department in the Enabling Computing Technologies Research Domain of Bell Labs. Her interests are in pattern recognition, data mining, and computational modeling and simulation. She pioneered research in decision combination in multiple classifier systems, random decision forests, data complexity analysis, and many topics in image and text analysis. She has also led major efforts on modeling and monitoring large-scale optical transmission systems. Recently she worked on problems in user profiling, optical network diagnostics, and customer experience management. She is Editor-in-chief of the journal Pattern Recognition Letters, and elected Fellow of the International Association for Pattern Recognition and the IEEE. She has over 90 publications and was granted 7 U.S. patents. She received a Ph.D. in Computer Science from SUNY at Buffalo in 1992.

PLACE PHOTO HERE

Biplab Sikdar (S'98, M'02) received the B. Tech degree in electronics and communication engineering from North Eastern Hill University, Shillong, India, the M. Tech degree in electrical engineering from Indian Institute of Technology, Kanpur and Ph.D in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA in 1996, 1998 and 2001, respectively. He is currently an Associate Professor in the Department of Electrical, Computer and Systems Engineering of Rensselaer Polytechnic Institute, Troy, NY, USA. His research interests include wireless MAC protocols, network routing and multicast protocols, network security and queueing theory. Dr. Sikdar is a member of IEEE, Eta Kappa Nu and Tau Beta Pi and is an Associate Editor of the IEEE Transactions on Communications.