A Statistical Approach to Anomaly Detection in Interdomain Routing

S. Deshpande[1], M. Thottan[2], T. K. Ho[2], B. Sikdar[1]
[1] Rensselaer Polytechnic Institute, Troy, NY 12180.
[2] Bell Laboratories, Murray Hill, NJ 07974.
Email: {deshps,sikdab}@rpi.edu, {marinat,tkh}@research.bell-labs.com

***Abstract*— A number of events such as hurricanes, earthquakes, power outages can cause large-scale failures in the Internet. These in turn cause anomalies in the interdomain routing process. The policy-based nature of Border Gateway protocol (BGP) further aggravates the effect of these anomalies causing severe, long lasting route fluctuations. In this work we propose an architecture for anomaly detection that can be implemented on individual routers. We use statistical pattern recognition techniques for extracting meaningful features from the BGP update message data. A time-series segmentation algorithm is then carried out on the feature traces to detect the onset of an instability event. The performance of the proposed algorithm is evaluated using real Internet trace data. We show that instabilities triggered by events like router mis-configurations, infrastructure failures and worm attacks can be detected with a false alarm rate as low as 0.0083 alarms per hour. We also show that our learning based mechanism is highly robust as compared to methods like Exponentially Weighted Moving Average (EWMA) based detection.**

I. Introduction

Stability of interdomain routing is of critical importance to maintain the connectivity and reliability of the Internet. However, interdomain exchange of traffic is between different administrative domains which makes the process of routing highly dependent on the local rules of these domains. Fortunately not all route changes can cause instability. Examples of events that do result in anomalous route changes are infrastructure failures (for instance due to disasters like hurricanes or earthquakes), power outages (large scale events like the blackout in North-Eastern US in August 2003), worm attacks and BGP router mis-configurations. Such anomalous route changes and the impact of local rules on these route changes can be observed by monitoring the BGP update messages seen at the peering points. However, monitoring BGP updates is a challenging task since there are multiple prefixes that need to be monitored [4]. The goal of this work is to detect the deviations from normal BGP update traffic with the purpose of identifying a small set of alarms that requires the close attention of a network provider.

There has been a flurry of recent work focussing on the detection of routing anomalies using BGP update message data. In [4], the authors propose a system that can be used for online generation of routing disruption reports. However, the system focusses on identifying events that originate close to the observation point and thus may not be effective in detecting wide-spread instabilities far from their observation point. The learning-based approach described in [2] proposes the use of wavelet transformations to extract the temporal patterns of BGP update-dynamics, which translates the problem into the wavelet domain. There is some concern about the loss of time granularity as a result of requiring a good sample support for accurate estimation of the wavelet basis. The methods in [16] and [10] utilize visual based techniques for the detection and location of the instabilities. In this approach the authors use data mining techniques to render the data free of noise and translate it into graphical views for easier identification by a human operator. As suggested by the authors, the visual detection approach could be used in conjunction with the automated detection scheme that is proposed in this paper. We use an online statistical time series based approach for detecting the occurrence of anomalies in BGP route update messages. We focus on building a detection mechanism that can be implemented on a single BGP router i.e. without the need for a distributed infrastructure.

Our approach is to perform a time domain analysis of features extracted from BGP update message data and use a learning-based algorithm for robust detection. We use filtering techniques to smooth noisy traces and then use adaptive segmentation techniques to capture abnormalities in the data. We also utilize the correlated presence of abnormalities across several features to reduce the occurrence of false positives in the detection. We use features that efficiently exhibit a distinct pattern of behavior during any anomalous period irrespective of the root cause event. This ensures accurate performance of the detection algorithm for many different kinds of events, independent of the training dataset.

The detection of the onset of an instability event is the first step towards isolating an anomalous period, after which effective root cause analysis can be performed. Several schemes in the past have proposed methods that modify the message handling procedures in BGP in order to limit its path exploration process [8][9][14][19] and enforce correct behavior [6]. We believe that these methods will be highly effective if implemented as emergency correction techniques when periods of large scale instabilities are seen. This will ensure that these mechanisms do not interfere with the normal operation of BGP. Our algorithm can be used in conjunction with these and similar techniques to detect and correct the onset of such periods of instabilities.

The rest of the paper is organized as follows: Section II describes the features, Section III explains the detection

scheme and Section IV explains the parameter estimation process. Sections V evaluates our proposed mechanisms using real BGP data and also presents a comparative study of our results with a variant of another detection approach [12]. Finally, Section VI discusses the important contributions and presents concluding remarks.

II. Feature Selection

The route fluctuations caused by any failure event cause an immediate effect on the route advertisement and message exchange operations of a BGP router. The update messages exchanged between routers directly reflect the effect of any anomaly. Hence, their contents form the data for our detection scheme. In order to ensure accurate detection we need those *features* of this data that show distinct behavior during normal and anomalous periods. We extract several features (described below) from the BGP update messages and identify the useful ones using scatter plots in Mirage [11]. The selected features are used in the form of a time series (or trace) collected every 5 minutes. We also use median filtering in order to smooth out any unwanted transients in the feature traces.

A. Message Volumes

Interdomain routing instabilities are characterized by a sharp and sustained increase in the number of announcement and withdrawal messages exchanged by many of the BGP routers [22] [13][7]. The Slammer worm attacked the Internet on the 25^{th} Jan 2003 [13]. On Oct 7^{th} 2001, AS2008 and AS3300 leaked private AS numbers from their confederation space due to a misconfiguration on their BGP routers [28]. A large peak in the number of announcements and withdrawals received at a router from its peers was co-incidental with the duration of these events. Figure 1 shows this effect.

B. AS Path Length

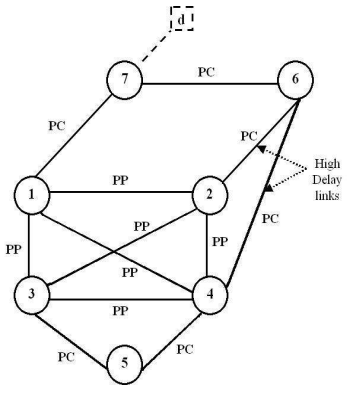


Fig. 2. Example topology with 7 AS nodes. The destination d is withdrawn leading to a sequence of update messages exploring different paths before the topology converges. Here, P-P implies peer-to-peer, P-C implies provider-to-customer relationship.

In case of any failure, the BGP path exploration process is triggered for the failed routes. As a result every router involved will try to search for possible alternate paths to the destination until either the path is completely withdrawn or a valid back up path is established and the topology converges.

For example, consider the topology in Figure 2. Each node represents a different AS with a single BGP router. We consider paths for the destination prefix "d" advertised by the router in AS7. The link labels denote the peering relationship between two nodes in their numeric order. For example, link $[6-7]$ is a [P-C] link indicating that AS6 is a provider for customer AS7. We assume the shortest path first policy at all nodes with a tie resolved according to the local rules. For example, AS2 always prefers routes learnt from AS3 over those learnt from AS4, if both the routes are of the same length. Note that the links between AS6 and its providers, AS4 and AS2 are high delay links.

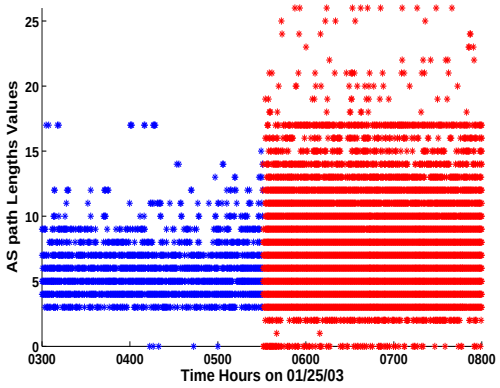


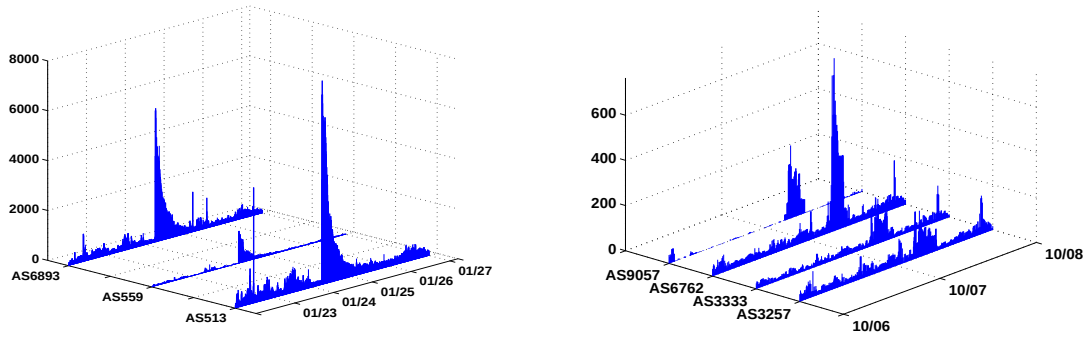
Fig. 3. The plot of number of messages with different AS path lengths received from the router in AS513 for 5 hours around the onset of the Slammer worm attack on 25th Jan 2003. The right half (red) indicates the time period after the attack. There is a sudden increase in the number of messages with AS path lengths greater than 4 or 5 i.e. the normal value of AS path length, at the onset of the worm (midpoint of the interval).

Table I shows the sequence of update messages exchanged between nodes after the path to destination "d" is withdrawn by AS7 due to some failure. The first stage lists the conditions after withdrawals sent by AS7 are received at AS6 and AS1. A stage is defined by the receipt and processing of one or more messages and transmission of resulting route updates by any node. We do not show the entire process (8 stages) till convergence, but just the initial five stages for illustration.

Table I shows that the lengths of AS paths received at most of the AS nodes increase at successive stages. For instance, the sequence of AS paths received at node 2 is: from AS1 - [1-7; 1-4-6-7], AS3 - [3-1-7; 3-4-6-7] and AS4- [4-6-7; 4-3-1-7]. Thus, the length of the AS paths received for the same destination changes from 2 to 4 hops. This effect is more prominent in the Internet due to high connectivity of the routers (ref. Figure 3).

We call the mode values of the distribution of AS path lengths as the "normal value of AS path length" and denote it by nv_l. The number of messages received with AS path lengths differing from this normal value is negligible during normal periods of operation but shows a prominent increase under instability conditions (ref. Figure 3).

Another reason for the receipt of routes with abnormally large AS path lengths upon failure events is AS path prepending that is very commonly used in the Internet to achieve traffic engineering [3]. AS path prepending is when 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 it less attractive to the BGP peers that base their route



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

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

Fig. 1. Message volumes for two periods of instability

TABLE I

THE INITIAL 4 STATES OF THE AS NODES IN THE TOPOLOGY OF FIGURE 2. THE FIRST STAGE BEGINS WITH THE RECEIPT OF WITHDRAWALS FOR DESTINATION "d" SENT BY AS7. THE AVAILABLE AS PATHS AT A NODE ARE LISTED IN ORDER OF PREFERENCE.

AS #	AS paths received	All available Paths to d	AS paths sent
Stage I			
AS6	AS7 ← w	None	w → AS4, AS2
AS5	None	[4-6-7d, 3-1-7d]	None
AS4	None	[6-7d, 1-7d, 3-1-7d, 2-6-7d]	None
AS3	None	[1-7d, 4-6-7d, 2-6-7d]	None
AS2	None	[6-7d, 1-7d, 3-1-7d, 4-6-7d]	None
AS1	AS7 ← w	[4-6-7d, 2-6-7d]	1-4-6-7d → AS3, AS2; w → AS4
Stage II			
AS4	AS1 ← w	[6-7d, 3-1-7d, 2-6-7d]	None
AS3	AS1 ← 1-4-6-7d	[4-6-7d, 2-6-7d, 1-4-6-7d]	3-4-6-7d → AS1, AS2, AS5; w → AS4
AS2	AS1 ← 1-4-5-7d	[6-7d, 3-1-7d, 4-6-7d, 1-4-6-7d]	None
Stage III			
AS4	AS6 ← w	[3-1-7d, 2-6-7d]	4-3-1-7d → AS2, AS5, AS6, AS1; w → AS3
AS2	AS6 ← w	[3-1-7d, 4-6-7d, 1-4-6-7d]	2-3-1-7d → AS1, AS2, AS4, AS6; w → AS3
Stage IV			
AS5	AS3 ← 3-4-6-7d	[4-6-7d, 3-4-6-7d]	None
AS4	AS3 ← w	[2-6-7d]	4-2-6-7d → AS3; w → AS2; MRAI → AS1, AS6, AS5
AS2	AS3 ← 3-4-6-7d	[4-6-7d, 3-4-6-7d, 1-4-6-7d]	2-4-6-7d → AS3, w → AS4, MRAI → AS1, AS6
AS1	AS3 ← 3-4-6-7d	[4-6-7, 2-6-7d];	None
Stage V			
AS5	AS4 ← 4-3-1-7d	[4-3-1-7d, 3-4-6-7d]	None
AS4	AS2 ← 2-3-1-7d	[2-3-1-7d]	w → AS2; MRAI → AS1, AS3, AS5, AS6
AS3	AS2 ← w	[4-6-7d, 1-4-6-7d]	None
AS2	AS4 ← 4-3-1-7d	[3-4-6-7d, 4-3-1-7d, 1-4-6-7d]	2-3-4-6-7d → AS4; w → AS3; MRAI → AS1, AS6

selection on the shortest path length criteria. As a result, these routes are the very rarely used backup routes. During a failure, these routes are also eventually selected when all other shorter routes fail and so, 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 might not necessarily be a unique list of AS numbers.

We use separate traces corresponding to each observed value of the AS path length as a feature trace. Thus, the AS path length feature set can be defined as:

$$ASPL = \{\bar{X_{ij}} = \langle x_0, x_1, \cdots \rangle; i = 1, \cdots, M_l; \quad (1)$$
$$j = 1, \cdots, NP\}$$

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.

C. 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 later effect by treating AS paths received in consecutive messages (for the same prefix) as strings and obtaining edit distances [26] 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.

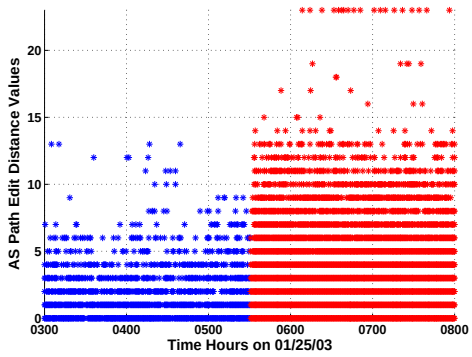


Fig. 4. The plot of number of messages with different pairwise AS path edit distances received from the router in AS513 for 5 hours around the onset of the Slammer worm attack on 25th Jan 2003. The right half (red) indicates the time period after the attack. At the onset of the attack (midpoint of the interval) there is a sudden increase in the number of successive messages with AS path edit distances greater than 0 or 1 i.e. the normal value.

We denote the mode of the AS path edit distance value distribution as nv_{ed} i.e. the "normal value for AS path edit distance". During an instability event, as all possible paths for a particular prefix are exchanged, a large number of successive messages show higher edit distances (more than nv_{ed}). Figure 4 shows this effect for the Slammer worm attack. In order to capture the effect of the instability on the AS path edit distance feature we use separate feature traces corresponding to each observed value.

Thus, the AS path edit distance feature set is defined as:

$$ASPED = \{\bar{X_{ij}} = \langle x_0, x_1, \cdots, \rangle; i = 1, \cdots, M_{ed}; j = 1, \cdots, NP\} \quad (2)$$

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

D. Relevant Features

After exploring the features mentioned above, our available feature set is:

$$F' = [AV, WV, ASPL, ASPED] \quad (3)$$

where, AV and WV are the volume feature sets i.e. the time series of the number of announcements and withdrawals received from each peer respectively, $ASPL$ is the AS path length and $ASPED$ the AS path edit distance feature set.

Since the maximum values of AS path length (M_l) and edit distance (M_{ed}) observed can be very high, the dimensionality of this feature set can also be very high. Hence, we need to use some filtering method to retain only highly discriminatory and relevant feature traces to be used for detection.

We discard the announcement feature trace as it shows a very high correlation with the AS path length feature traces. We also filter out the feature traces corresponding to the normal AS path length (nv_l) and normal AS path edit distance (nv_{ed}) values. This is done since the prime observation that characterized the anomalous behavior pattern was the increase in number of messages with "ab"-normal values of the features. The features are then selected based on their discrimination capability or feature efficiency that is calculated using a method based on the Fisher's Linear Discriminant [5], [21]. The value of nv_l observed for all of the datasets we use is in the range (4, 5) and the value of nv_{ed} in the range (0, 1).The values for nv_l and nv_{ed} can vary for different BGP routers and are selected by observing the behavior of the edit distance traces received by the router for an extended period of normal operation.

Thus, for data from each peer the final feature set that we use for instability event detection has 9 traces:

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

III. Detection Algorithm

The detection scheme we use is based on adaptive sequential segmentation [25]. The core of the segmentation is change detection using a Generalized Likelihood Ratio (GLR) based hypothesis test. We give a detailed description of the GLR technique in Section III-A and then follow with the various steps associated with the algorithm used for overall segmentation in Sections III-B, III-C and III-D.

A. Generalized Likelihood Ratio Test

We will give the details of the basic GLR test used for change detection in this section. The non-stationary feature time series is represented in terms of piecewise stationary segments of data called the learning and test windows.

Thus, consider a learning window $L(t)$ and test window $S(t)$ of lengths N_L and N_S respectively. They can be represented as:

$$L(t) = \{l(t_1), l(t_2), \cdots, l(t_{N_L})\}$$
$$S(t) = \{s(t_1), s(t_2), \cdots, l(t_{N_S})\} \quad (4)$$

Any $l(t_i)$ (or $s(t_i)$) in the equation above can be expressed as $\tilde{l}(t_i)$ where $\tilde{l}(t_i) = l(t_i) - \mu$ and μ is the mean of the segment $L(t)$. Now, $\tilde{l}(t_i)$ can be modeled as an auto-regressive (AR) process of order p with a residual error $\epsilon(t_i)$

$$\epsilon(t_i) = \sum_{k=0}^{p} \alpha_{lk} \tilde{l}(t_i - k) \quad (5)$$

where $\alpha_L = \{\alpha_{l1}, \alpha_{l2}, \cdots, \alpha_{lp}\}$ and $\alpha_0 = 1$ are the AR parameters. Assuming each residual time is drawn from an $N(0, \sigma_L^2)$ distribution, the joint likelihood of the residual time series for the learning window is given by

$$p(\epsilon(t_{p+1}), \cdots, \epsilon(t_{N_L}) | \alpha_{l1}, \cdots, \alpha_{lp}) = \left(\frac{1}{\sqrt{2\pi\sigma_L^2}}\right)^{\acute{N}_L} e^{\left(\frac{-\acute{N}_L \hat{\sigma}_L^2}{2\sigma_L^2}\right)} \quad (6)$$

where σ_L^2 is the variance of the segment $L(t)$, $\acute{N}_L = N_L - p$ and $\hat{\sigma}_L^2$ is the covariance estimate of σ_L^2. Similarly, the test window is also modeled using AR parameters, $\alpha_S = \{\alpha_{s1}, \alpha_{s2}, \cdots, \alpha_{sp}\}$ and $\alpha_0 = 1$, σ_S^2 = the variance of the

segment $S(t)$, $\acute{N}_S = N_S - p$ and $\hat{\sigma}_{S^2}$ = the covariance estimate of σ_S^2.

Thus, the joint likelihood ν of the two segments $L(t)$ and $S(t)$ is given by

$$\nu = \left(\frac{1}{\sqrt{2\pi\sigma_L^2}}\right)^{\acute{N}_L} \left(\frac{1}{\sqrt{2\pi\sigma_S^2}}\right)^{\acute{N}_S} e^{\left(\frac{-\acute{N}_L\hat{\sigma}_L^2}{2\sigma_L^2}\right)} e^{\left(\frac{-\acute{N}_S\hat{\sigma}_S^2}{2\sigma_S^2}\right)} \tag{7}$$

This likelihood ν is used to perform a binary hypothesis test based on the Generalized Likelihood Ratio. Under the hypothesis H_1 implying that a change is observed between the two windows, we have $\alpha_L \neq \alpha_S$ and $\sigma_L^2 \neq \sigma_S^2$.

Then under H_1 the likelihood becomes:

$$\nu_1 = \nu \tag{8}$$

and under H_0 the likelihood becomes:

$$\nu_0 = \left(\frac{1}{\sqrt{2\pi\hat{\sigma}_P^2}}\right)^{\acute{N}_L+\acute{N}_S} e^{\left(-\frac{(\acute{N}_L+\acute{N}_S)\hat{\sigma}_P^2}{2\sigma_P^2}\right)} \tag{9}$$

where $\hat{\sigma}_P^2$ is the pooled variance of the learning and test windows. Using the maximum likelihood estimates for the variance terms in Equations 8 and 9, the likelihood ratio is therefore, given by

$$\eta = \frac{\nu_0}{\nu_1} = \hat{\sigma}_P^{(\acute{N}_L+\acute{N}_S)} \hat{\sigma}_L^{-\acute{N}_L} \hat{\sigma}_S^{-\acute{N}_S}. \tag{10}$$

For computation purposes we use the logarithmic form of the above equation given as:

$$d = (\acute{N}_L + \acute{N}_S)\ln(\hat{\sigma}_P^2) - (\acute{N}_L \ln\hat{\sigma}_L^2 + \acute{N}_S \ln\hat{\sigma}_S^2) \tag{11}$$

We refer to d as described above to be the GLR distance between the two windows. In the GLR test, then, d is compared to a reasonably chosen threshold δ to determine whether the two windows are statistically similar or not.

B. Detection of Segment boundaries

Segment boundary detection isolates periods of abnormal behavior seen in the feature traces. The boundary points detected here are points where the behavior of the message traces deviates significantly from the period since the last segment boundary was detected. The GLR technique is used for sequential segmentation of the feature traces by imposing learning and test windows. Consider that for a feature time series, the segmentation algorithm has most recently detected a segment boundary at an arbitrary time index $t = r$; without loss of generality we can define $r = 1$. The decision process necessary to detect a new boundary at an arbitrary time index $s > L$ (L is the minimum segment length) then, is 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 a predetermined threshold δ. At this point, the time index $s+L-1$ is called the "detection time" t_D.

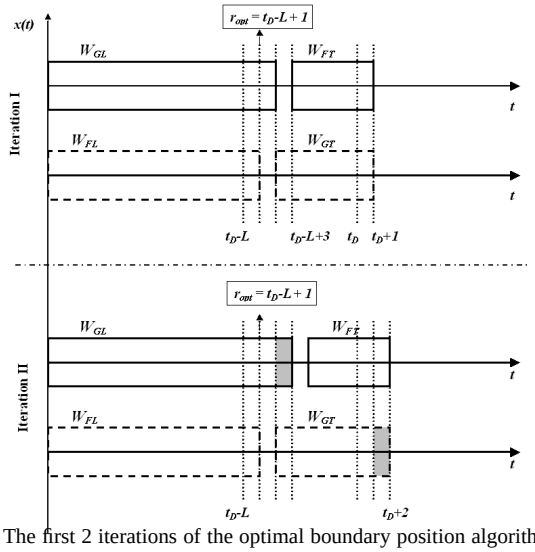


Fig. 5. The first 2 iterations of the optimal boundary position algorithm. The shaded regions of W_{GL} and W_{GT} indicate their growth in the next iteration. The second iteration shown is under the assumption that the GLR distance $d(W_{GL}, W_{FL})$ is less than $d(W_{GL}, W_{FT})$ and the boundary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

C. Location of Optimal Boundary Position

The main purpose of this step is to detect the exact location of the change point in the traces. This exact boundary position can be anywhere within the range $(t_D - L + 1, \cdots, t_D)$. This step involves using different combinations of test window and learning window sizes to detect the position where the maximum change occurs between the two windows. Hence the name 'optimal boundary position'.

We now describe this process in detail. Initially the optimal boundary position 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 window (W_{GT}). The initial windows are (ref. Figure 5):

$$\begin{aligned} W_{GL} &: \langle x(1), \cdots, x(t_D - L + 2)\rangle, \\ W_{FT} &: \langle x(t_D - L + 3), \cdots, x(t_D + 1), \\ W_{FL} &: \langle x(1), \cdots, x(t_D - L + 1)\rangle, \\ W_{GT} &: \langle x(t_D - L + 2), \cdots, x(t_D + 1)\rangle \end{aligned} \tag{12}$$

The growing window sizes increase and the fixed test window moves ahead by one at each iteration. Note, that the total length composed of both windows is identical in both cases and grows continuously. 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 between the GLR distances between these two pairs of windows.

When the last potential boundary position is reached, the algorithm stops and the last allocated boundary position is the optimized boundary. Thus, at the end of the last iteration the learning window size grows from $t_D - L + 2$ to t_D.

For any general case, the final 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 dependant on the

Algorithm 1 Change detection and optimal boundary location.

$s = L$; set s as the end of the learning window and start of the 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 the current test window
  for ($t_D - L + 2 \leq s \leq t_D$) **do**
    $g_1 = d(s, s+L-1)$; GLR distance between the growing learning-fixed test windows
    $g_2 = d(r, s+L-1)$; GLR distance between the fixed learning-growing test windows
    if $(g_1 > g_2)$ **then**
      $r = s$; detected a better boundary position
    end if
    $s = s + 1$;
  end for
  $ropt = r$; optimal boundary position found
  $data = [data(r) : data(sizeof(data))]$; further segmentation on remaining data
end while
Note: L: minimum initial window size; δ: GLR threshold;
$d(x, y)$: GLR distance between windows $[1, x]$ and $[x+1, y]$.

Algorithm 2 Processes for online alarm clustering.

n; number of traces = 2 * number of peers
a_i; alarm for trace i, $i = 1, 2, ..., n$
$t(a_i)$; time at which alarm a_i occurs
τ; maximum time interval between alarms in the same cluster
$N(a_i)$; The cluster of alarms in the neighborhood of alarm a_i
A; Final alarm indicating an instability event

FOR EVERY ALARM a_i
Set a timer for τ; A timer that inactivates the alarm at after τ
Set $N(a_i) = a_i$; include the alarm in its own neighborhood
if (a_j still active and $i \neq j$) **then**
  Include a_i in $N(a_j)$; include alarm a_i in the neighborhood of other active alarms
end if

SUBROUTINE FOR TIMER EXPIRATION
if $(N(a_i))$ **then**
  Delete $N(a_i)$; Delete the neighborhood of the alarm
  Delete a_i from all $N(a_i)$; Delete the alarm from any other neighborhood
end if
Delete a_i;

BACKGROUND PROCESS
if $(|N(a_i)| \geq \frac{n}{2}$) **then**
  $A = TRUE$; A significant instability event detected
  Delete $N(a_i)$; Delete the alarm and its neighborhood
  Delete a_i from all $N(a_j)$;
  Delete a_i;
end if

minimum window size L. Though, locating the optimal boundary position introduces a delay in detection, it is important for the main purpose of avoiding any false alarms. In Algorithm 1 we present the pseudo-code for the change point detection and optimal boundary location steps. The segmentation of each of the feature traces from the set F' is carried out using the procedure described above and the allocated boundary position at the end of the optimal boundary location is said to be the time instant of the onset of an instability according to that particular trace for that particular peer.

D. Alarm Correlation

The change detection and optimal boundary location processes are applied to each feature trace from each peer and the change points detected at the end are termed as per-feature-trace alarms. In order to make the detection process more robust against volatility of feature traces, the final instability indicator is a combination of these alarms. These combinations are built using a two step process:

Step I: The alarms from the feature traces for the different values of AS path lengths (or AS path edit distances) are clustered in time using the complete linkage algorithm [23]. We implement this online as per Algorithm 2. We define $N(a_i)$ as the neighborhood of any alarm generated by the i^{th} trace. Also associated with each alarm a_i we have a timer that 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

TABLE II

TRAINING(*) AND TEST DATASETS FOR DIFFERENT INSTABILITIES. TRAINING DATA IS USED FOR FEATURE SELECTION AND PARAMETER ESTIMATION. NOTE THAT THE RRCS LISTED HERE MAINTAIN BGP SESSIONS OVER DIRECT CONNECTIONS WITH THE PEERS AND NOT MULTI-HOP BGP SESSIONS.

Data for	Month	RRC	Number 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
Misconfiguration error*	October 2001	rrc03, Amsterdam	4	AS3257, AS3333, AS6762, AS9057
Nimda Worm*	September 2001	rrc04, Geneva	3	AS513, AS559, AS6893
Code Red II Worm	July 2001	rrc04, Geneva	3	AS513, AS559, AS6893
Moscow Blackout	May 2005	rrc04, Geneva	3	AS513, AS559, AS20932

are 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 background process keeps track of all the alarm neighborhoods and as soon as any neighborhood contains $\lceil n/2 \rceil$ 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:* We cluster the alarms generated by AS Path Length, AS Path Edit Distance at end of step I and the change points detected by the Withdrawal traces in time, again using complete linkage. If the cluster strength is 2 or more elements, an instability-alarm is generated (ref. Figure 6). The different pairs of features are preserved in the combination scheme for possible classification of different kinds of instabilities.

These steps are implemented separately on data from each peer and so the final alarms are generated on a per peer basis.

E. Preventing False Alarms

The use of AS path length and AS path edit distance helps to lower the false alarm rate and minimizes detection delay by maintaining a low clustering threshold τ. This is a significant advantage over using just volume-based detection.

In order to ensure that we detect the instabilities correctly and do not miss any, we need the individual feature trace alarms to be as precise as possible. This is done by locating the optimal boundary in the segmentation process. We could increase the cluster threshold (τ) in order to cluster a delayed change point, but this can lead to an increased number of false alarms and delay the detection of the instability.

IV. PARAMETER ESTIMATION

The performance of the detection algorithm depends on the values of several parameters. The feature traces are median filtered in order to avoid capturing transient peaks. The order of the median filter is set as $m = 7$ to suppress all peaks upto 15 minutes. The initial window size L is chosen as 20 as it has to be at least twice as large as m, to avoid using a window that is entirely smoothed out. The order of the AR process is selected on the basis of the Akaike's Information Criteria (AIC) [27]. The GLR threshold value δ is learnt from the data during the normal periods. The clustering threshold is chosen empirically as 50 minutes, so that the final alarm is generated within an hour of the effect of the instability being seen in the feature traces. We use datasets corresponding to 3 different types of events as training data to estimate these parameter values. It is important to note that while evaluating the detection algorithm, the same parameter values are maintained irrespective of the anomaly event and dataset used.

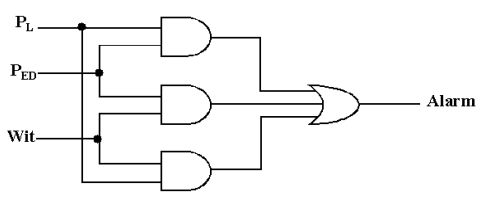


Fig. 6. The combination of the withdrawal (Wit), ASpath length (P_l), and ASpath edit distance (P_{ed}) feature alarms that is used to generate the final instability indicator alarm. The "AND" is TRUE if 2 alarms are in one cluster.

V. EVALUATION OF THE DETECTION SCHEME

In order to validate the detection mechanism we use data collected at the Réseaux IP Européens (RIPE) remote route collectors (RRC's). We use the data from only those route collectors that have a direct BGP connection with peers located at the same exchange points to avoid any impact of the collection process itself on the data [20]. We use five-day long traces for testing the detection of several types of well-known anomaly events. The raw data collected is in the form of update message logs received during 5 minute intervals (or we convert it to 5 minute intervals). For each period of instability, we use data from a different route collector. This is done because some events have a significant impact only at specific RRCs. This also tests the scheme for data from diverse sources. Table II gives a list of the data traces we use for the different instability types. We observe that even though the RRCs listed in the table peer with many more BGP routers [15], only the peers mentioned in the table send a statistically significant amount of update messages during the particular period we tested on. We gather information about anomalous events from the North American Network Operators group mailing list [28].

We use some of the datasets as training data for feature selection and parameter estimation. The training data comprises data from several anomaly periods and different sources. Specifically, we use data from durations corresponding to a worm attack (Nimda worm, Sep 2001), a failure event (Moscow Blackout affecting the MSIX, May 2005) and a BGP router misconfiguration event (Export misconfiguration in AS3300, AS2008, Oct 2001). For each event we have data from different peers of one of the RIPE RRCs (cf. Table II).

The rest of the datasets in Table II are used as test data. We emphasize here that the training data and test data are mutually exclusive in terms of events, time periods and also source RRCs. The performance of the algorithm is presented in Table III. Also, Figure 7 shows the results for all stages

TABLE III

THE RESULTS FOR OUR DETECTION ALGORITHM. THE TOTAL NUMBER OF FALSE ALARMS USING THE EWMA SCHEME ON VOLUME BASED FEATURES IS 111; WITH THE EWMA SCHEME ON THE VOLUME BASED AND AS PATH BASED FEATURES IS 36; USING OUR SCHEME IS JUST 6. THE NUMBER OF ALARMS IN THE TRUE ALARM CLUSTER FOR THE EWMA-BASED SCHEMES INDICATE THE DIFFICULTY IN USING A PERSISTENCE BASED APPROACH TO FILTER OUT FALSE ALARMS. 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.

Event	Slammer Worm			Moscow Blackout VIX dataset			BGP Misconfiguration		
Month	Jan 2003			May 2005			Oct 2001		
AS Num.	AS513	AS559	AS6893	AS1853	AS12793	AS13237	AS3257	AS6762	AS3333
Duration*	22hrs	21hrs.	30hrs	6hrs.	8hrs	<2hrs.	5hrs	7hrs.	6.5hrs
Our Detection Algorithm									
# Alarms in the true alarm cluster	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**1**
# False Alarms	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
EWMA-mechanism using only volume features									
# Alarms in the true alarm cluster	246	254	271	2	0	47	4	72	28
# False Alarms	6	14	6	2	47	0	8	2	8
EWMA-mechanism using all our features									
# Alarms in the true alarm cluster	209	134	156	0	1	0	0	68	14
# False Alarms	1	12	4	0	1	0	1	1	3

Event	Nimda Worm			Code Red II Worm			Moscow Blackout CIXP dataset		
Month	Sep 2001			July 2001			May 2005		
AS Num	AS513	AS 559	AS6893	AS513	AS559	AS6893	AS513	AS559	AS20932
Duration*	>48hrs	>48hrs	>48hrs	11hrs	11hrs	11hrs	16hrs	<1hr.	4hrs.
Our Detection Algorithm									
# Alarms in the true alarm cluster	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**
# False Alarms	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**1**
EWMA-mechanism using only volume features									
# Alarms in the true alarm cluster	77	20	236	0	20	13	15	6	78
# False Alarms.	3	10	3	2	11	4	6	6	12
EWMA-mechanism using all our features									
# Alarms in the true alarm cluster	0	0	153	0	14	6	0	0	48
# False Alarms	1	3	1	0	2	0	0	3	3

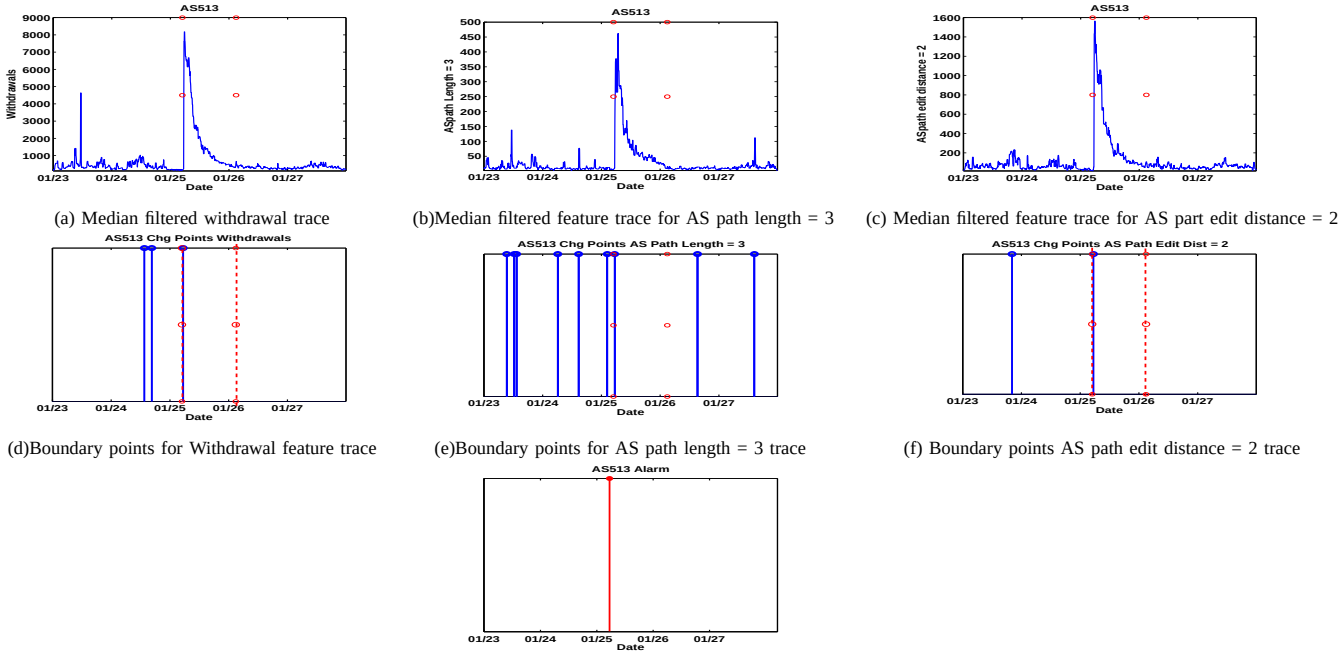
of alarm detection for the Slammer worm attack in Jan 2003 and Figure 8 shows a similar plot for detection of the Moscow Blackout event in May 2005.

Table III shows that our algorithm detects the occurrence of most anomalous events and has very few false alarms. For some of the events, the algorithm does not generate an alarm for every peer. These alarms can be deemed as missed alarms. These missed events can also be detected when the algorithm is trained on data for a longer period.

A. Comparison with EWMA-based detection

We implement an EWMA based detection scheme in order to check its effectiveness for anomaly detection as compared to our algorithm. We use the adaptive EWMA scheme described by Griffin et.al. [12] to accommodate any linear trends or baseline shifts in the feature time series. In this work, the authors use 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. So for comparison, we have implemented the EWMA scheme on the message volume features (i.e. announcement and withdrawal feature traces received from each peer). In our implementation, the final alarm is generated only after the alarms generated by the announcement and withdrawal traces separately are combined using an 'AND' operation. We implement a logical 'AND' here and do not allow any time threshold for clustering alarms to make the alarm generation more constrained. Also, in order to get comparable results between the two mechanisms, we use median filtered traces even for the EWMA-based detection with the same filter order (7).

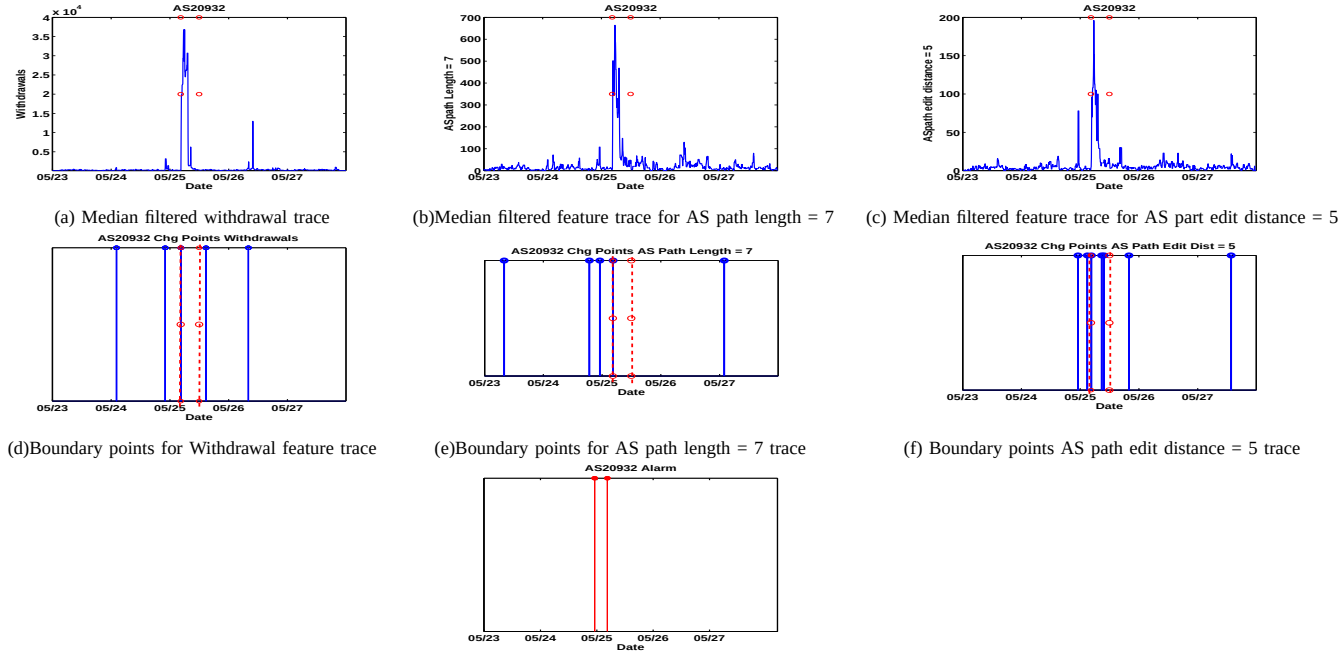
The EWMA detection scheme using volume-based features generates a large number of false alarms (cf. Table III). Overall, the number of false alarms generated is 111 (for 6 datasets covering 5 events and 5 days per event), whereas our algorithm generates only 6 false alarms for the same datasets and period. Each alarm generated by the EWMA scheme is actually a cluster of a large number of contiguous alarms. Therefore, a persistence based mechanism can be used to filter out the false alarms. However, the number of alarms in the cluster of true alarms is highly variable (cf. Table III). Thus, it is extremely difficult to obtain an optimal value of the cluster size (i.e. persistence of the alarms). Also, using such a persistence based scheme to isolate only the true alarms will lead to an increased detection delay for the EWMA-based algorithm. For instance if we assume a persistence value of say 50 i.e. the alarm is considered valid only when 50 contiguous alarms are obtained, the detection delay of the algorithm will become more than 20 hours which is highly impractical. Also, in this scenario the number of true alarms reduces along with the false alarms (only 7 of the 18 events from Table III will be detected). Our algorithm is capable of learning much better



(a) Median filtered withdrawal trace (b)Median filtered feature trace for AS path length = 3 (c) Median filtered feature trace for AS part edit distance = 2

(d)Boundary points for Withdrawal feature trace (e)Boundary points for AS path length = 3 trace (f) Boundary points AS path edit distance = 2 trace

Fig. 7. Results of the different steps of the algorithm for the Slammer worm Jan 2003. The data is shown for 23rd to 27th of Jan , CIXP data from peer in AS513; Parameters $L = 100$mins, $\rho = 1$, $\delta = 95^{th}$ quantile of the normal period GLR, $\tau = 50$mins. The red dashed lines and circular markers indicate the period of the Slammer worm. Figures(d),(e) and (f) show the alarms after optimal boundary location. All the other feature traces show a similar behavior during the anomaly period and several such boundary points are detected for each of those traces. The last figure shows the alarm obtained after correlating the alarms from all the feature traces.



(a) Median filtered withdrawal trace (b)Median filtered feature trace for AS path length = 7 (c) Median filtered feature trace for AS part edit distance = 5

(d)Boundary points for Withdrawal feature trace (e)Boundary points for AS path length = 7 trace (f) Boundary points AS path edit distance = 5 trace

Fig. 8. Results of the different steps of the algorithm for the Moscow blackout May 2005, rrc04 CIXP data from peer AS20932; Parameters: $L = 100$mins, $\rho = 1$, $\delta = 95^{th}$ Quantile of the Normal period GLR, $\tau = 50$mins. The red dashed lines and circular markers indicate the period of the blackout. Figures(d),(e) and (f) show the alarms after optimal boundary location. All the other feature traces show a similar behavior during the anomaly period and several such boundary points are detected for each of those traces. The last figure shows the alarm obtained after correlating the alarms from all the feature traces.

than the EWMA-based scheme. Thus, we do not need to rely on a persistence filter to obtain a low false alarm rate.

It is important to note however, that the EWMA scheme that we implemented relied on just BGP data whereas the authors in [12] use a combination of BGP and SNMP data to minimize the false alarms. In order to reduce the false alarms we also implement the EWMA scheme using all our features. The final alarm is thus generated only after the alarms from all the feature traces coincide. Using all the features the overall number of false alarms dropped down to 36. This is significantly lower than the EWMA-based results with simple volume features. However, with more constrained alarm generation, the number of missed alarms also increases (cf. Table III). We represent a missed alarm in Table III by a 0 in the field *number of alarms in the true alarm cluster*. Our detection algorithm performs significantly better in terms of

the false alarm rate and slightly better in terms of the missed alarms. However, with longer learning periods we expect much fewer missed alarms. These results indicate that a learning-based approach for detection performs much better than a adaptive threshold based EWMA approach.

VI. Conclusion

This paper demonstrates the efficacy of capturing the statistical changes in features extracted from BGP update message data for *online* detection of routing instabilities on a *single router*. The algorithm learns from the data itself without any prior assumptions about the fault model or the data distribution. We extract features based on three observations of the received update message data at any BGP router during almost all anomalous events:

- Sustained burst in message volume.
- Large amounts of rapid fluctuations in the route received for the same destination prefix.
- Receipt of uncommonly long backup AS paths.

A significant advantage of these features is that, our scheme can potentially detect large scale instabilities caused due to hitherto unseen events.

We use the temporal correlations amongst features effectively to minimize the number of false alarms. We have shown the validity of our approach through extensive evaluations with data during several kinds of events. We have shown that our scheme performs much better (false alarm rate of 0.00833 alarms/hr) in comparison with an EWMA-based approach (false alarm rate of 0.1541 alarms/hr not counting the large number of spurious alarms with the true alarms). A very low false alarm rate is significant since the detection of an anomaly event can be used to trigger a mechanism that can alter the route exchange process of BGP routers.

Our algorithm currently incurs an average detection delay of approximately 50 minutes. We are exploring alternative alarm combination schemes to reduce this delay. We also plan to incorporate pattern classification methodologies to perform root cause analysis after the detection of an anomaly event. We believe that our mechanisms can complement most of the existing work on improving BGP through changes to the message handling procedures.

References

[1] B. Zhang, D. Pei, D. Massey, and L. Zhang, "Timer Interaction in Route Flap Dampening," *In Procs. of the 25th International Conference on Distributed Computing Systems (ICDCS)*, pp. 393-403, Columbus, OH, June, 2005.

[2] J. Zhang, J. Rexford and J. Feigenbaum, "Learning-Based Anomaly Detection in BGP Updates," *In Procs. of the 2005 ACM MineNet Workshop*, pp. 219-220, Philadelphia, PA, Aug, 2005.

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

[4] J. Wu, Z. M. Mao, J. Rexford and J. Wang. "Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network," *In Procs. of Networked Systems Design and Implementation*, Boston, MA, May, 2005.

[5] J. Wang, X. Chen and W. Gao, "Online selecting discriminative tracking features using particle filter," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR* Volume 2, pp. 1037 - 1042, San Diego, CA, June, 2005.

[6] N. Feamster, "Practical Verification Techniques for Wide Area Routing," *ACM Sigcomm Computer Communication Review*, vol. 34, no. 1, pp. 87-92, January 2004.

[7] S. Deshpande, M. Thottan and B. Sikdar, "Early Detection of BGP Instabilities Resulting from Internet Worm Attacks," *In Procs. of IEEE GLOBECOM*, pp. 2266-2270, Dallas, TX, Nov, 2004.

[8] D. Pei, M. Azuma, D. Massey, and L. Zhang, "BGP-RCN: Improving BGP Convergence through Root Cause Notification," *Computer Networks and ISDN Systems,*, Vol. 48, no. 2, pp. 175-194, June 2005.

[9] D. Pei, M. Lad, D. Massey, and L. Zhang, "Route Diagnosis in Path Vector Protocols," *ACM Sigcomm Poster Session*, September, 2004.

[10] S. T. 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 Procs. of CCS Workshop on Visualization and Data Mining for Computer Security, ACM Conference on Computer and Communications Security,*' pp. 35-44, Washington, DC, Oct, 2004.

[11] T. K. Ho, "Mirage: Interactive Tools for Pattern Discovery," *In Procs. of the 17th International Conference on Pattern Recognition*, pp. 509-512, Cambridge, U.K., Aug, 2004.

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

[13] 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 Distrbuted Computing (IWDC)*, pp. 66-79, Kolkata, India, Dec, 2003.

[14] A. Bremler-Barr, Y. Afek and S. Schwarz, "Improved BGP Convergence via Ghost Flushing," *In Procs. of IEEE INFOCOM*, San Francisco, CA, March 2003.

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

[16] S. T. Teoh, K. Ma, S. F. Wu, D. Massey, X. Zhao, D. Pei, L. Wang, L. Zhang and R. Bush, " Visual-based Anomaly Detection for BGP Origin Change Events," *In Procs. of the 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, pp. 155-168, Heidelberg, Germany, October, 2003.

[17] M. Thottan and C. Ji, "Anomaly detection in IP networks,"*IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp.2191-2204, Aug, 2003.

[18] Z. Mao, R. Govindan, G. Varghese and R. Katz, " Route flap dampening exacerbates internet routing convergence," *In Procs. of the 2002 conference on Applications, technologies, architectures and protocols for computer communications*, pp. 221-233, Pittsburh, PA, Aug, 2002.

[19] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, S. Wu and L. Zhang, "Improving BGP Convergence through consistency assertions," *In Procs. of IEEE INFOCOM*, pp. 902-911, New York, NY, June 2002.

[20] 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," *in Proc. of ACM SIGCOMM Internet Measurement Workshop*, pp. 183-195, Marseille, France, Nov, 2002.

[21] T. K. Ho, Mitra Basu, "Complexity Measures of Supervised Classification Problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 289-300, March, 2002.

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

[23] R. Duda, P. Hart and D. Stork, "Pattern Classification, $2^n d$ Ed.," Jonh Wiley and Sons, 2001.

[24] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. "Delayed Internet Routing Convergence," *In Procs. of SIGCOMM*, pp.175-187, Stockholm, Sweden, Aug, 2000.

[25] U. Appel, A. Brandt, "Adaptive Sequential Segmentation of Piecewise Stationary Time Series," *Information Sciences* vol. 29, no. 1., pp. 27-56,1983.

[26] R. A. Wagner and M. J. Fisher, "The string to string correction problem," *Journal of Assoc. Comp. Mach.*, vol. 21, no. 1, pp. 168-173, Jan, 1974.

[27] H. Akaike, " Information theory as an extension of the maximum likelihood principle" *In Procs. of the Second International Symposium on information theory*, pp. 267-281, Budapest, Hungary, 1973.

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