Mining NANOG Mailing List*

Tony Zu-Cheng Huang, Chi-Yao Hong
Department of Computer Science
University of Illinois at Urbana-Champaign
{zuhuang1, hong78}@illinois.edu

ABSTRACT

It had been shown that the misbehaviors by few malicious, compromised or misconfigured BGP routers could lead to serious outages in Internet. This failing becomes progressively crucial as the recent prosper of outage-sensitive applications such as Voice over IP, streaming media, and video conferencing. To address these misbehaviors, previous work mainly focus on distributedly detect or prevent outages using limited state of Internet.

In this paper, we present a *first* step towards efficient troubleshooting by mining network operators mailing lists. Using Natural Language Processing (NLP) and Machine Learning, we develop a new approach to extract useful information from the mailing forum on North American Network Operators Group (NANOG).

Our experimental results show that the proposed approach detects 94 out of 105 outages from NANOG with a false positive rate of only 7.3%. We validate the extracted outage using real network logs collected by Route Views project. While our approach is not perfectly accurate, we envision it to be a useful information to existing anomaly detection/prevension mechanisms.

1. INTRODUCTION

Internet today adopts the Border Gateway Protocol (BGP) [20] as the core routing protocol to connect autonomous systems (ASs). However, the current design of BGP assumes that each border router will fully obey the rules specified by BGP, which is not always true in practice. The misbehavior such as faulty announcements could be sent by malicious, compromised or misconfigured routers, and could lead to serious problems. The common BGP problems include prefix filtering, prefix leaking, prefix hijacking, blackhole, and route instability [11].

*CS598PBG Project Report, Fall 2009.

Dealing with BGP problems is not new – different mechanisms have been proposed to mitigate the performance degradation induced by a variety of BGP problems [8, 13, 15, 21]. However, these mechanisms usually has complicated implementations and overheads in terms of control messages and implementation costs, and thus most of ISPs are not able to install most up-to-date mechanisms. Besides, these mechanisms might not be accurate (i.e., with considerable false positives/negatives) because of limited view of networks [16]. Due to the scalability issues, the BGP router does not have global view of entire Internet. The partial view of whole network renders the troubleshooting of network problems more difficult. Worse still, some problems, such as prefix hijacking, has been shown by recent researches to be hard to detect automatically [9].

In actual network management practice, each AS has network operators that manages the filter list, local transit policy and other network management issues. They also actively monitors network conditions and shares information about potential network problems. Network outages, illogical BGP announcements, prefix hijackings and other network problems are all topics closely monitored by these operators. Network operators communicate with each other through mailing lists and other channels. Therefore, instead of using existing mechanisms to deal with network outages, operators adopt various mailing lists as a good source of information on the history of network outages as well as a warning flag on current network problems.

In this paper, we try to leverage the information contained in the operator mailing lists and Route Views archive to help to solve the problem of network outages detection, diagnosis and analysis. We propose a platform to extract useful information from network forum. In particular, our platform can mine important information about present network problems from on-line forum. We here focus on the mailing forum on North American Network Operators Group (NANOG) [4], which is a public forum for the exchange of network operation issues for operators mainly within United States. Since most on-line forums adopt similiar design patterns, we believe our platform can be easily modified to support other forums, and we consider it as a first step towards efficient troubleshooting by mining Internet forums.

With the information extracted from NANOG, the current mechanisms could detect network problems more accurately. Extracted information from forum could help mechanisms

not only refine their searching domain (i.e., achieving fewer false negatives) but also improve the searching accuracy (i.e., having fewer false positives). While our approach is also not perfectly accurate, we envision it might be of use to signal "hint" to network operators regarding which AS, prefix or IP address may be faulty.

The main contributions of our design are as follows.

i. **Using information retrieval and natural language processing techniques to solve networking problem:** Troubleshooting network problem is hard for network operators because of the limited view of networks. We present a novel approach to diagnose network problem by mining the NANOG mailing lists. To the best of our knowledge, this paper is the first work to detect network anomalies by extracting on-line forum information.

ii. **Effective extraction:** Our design utilizes mature existing techniques from document indexing, classification and name entity recognition and combine them in an efficient way. It can be shown that our system effectively reports network problems that are useful for network problem diagnosis.

iii. **Classified report:** We mine the mailing list in a structured way, identify threads discussing network outages, classify them into outage categories and extract information about the outages. Our classified information can be used as error signal to network operators.

The proposed method is presented in §2, including thread classification and information extraction. We use statistical analysis and real network logs to evaluate the performance of our design in §3. We conclude our work in §4.

2. METHOD

Our design includes three stages as shown in Figure 1. In the first stage (§2.1) the raw information are extracted from NANOG and stored in a thread-level data structure. Given the thread-level information, we classified threads into network outage-related threads and others in the second stage (§2.2). Then the last stage (§2.3) extracts *useful* information from outage-related threads and report the extracted information for network diagnosis.

2.1 Raw Text Extraction and Processing

The input of this stage is the raw NANOG mailing list archive in HTML format, while the desired output is a list of *threads*. In forum structure, a thread consists of a collection of *posts*, which are user submitted messages. We use thread as the basic data structure to store the information extracted from NANOG, i.e., we classify posts into threads. Based on our observation, useful information for a single outage are scattered in various posts with the *same title*. Therefore, we use thread instead of post as the basic unit for our classifier and information extractor.

For each thread, we are particularly interested in extracting following features:

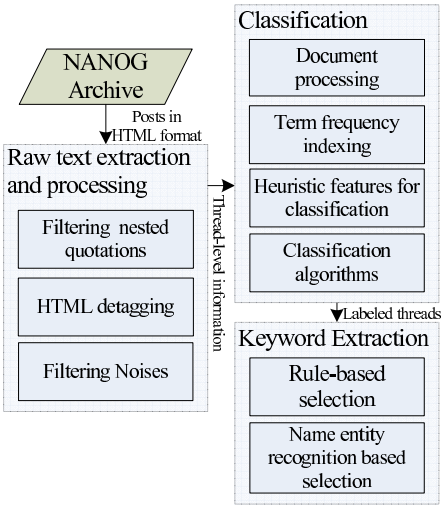

Figure 1: The architecture of the design.

- Title
- Author of the first post
- Publish time of the first post
- Publish time of the last post
- Post bodies (user messages)
- Keywords (the selection of keywords is discussed in §2.3)

We are interested in the authors of the posts because it would be useful to identity point of contact for information about the problem discussed in the thread. We reason that, in posts discussing network issues happening at real-time, the publish time of the first post might be close to the actual time when the outages are happening. We reason that, authors discussing about problem happened some time ago would explicitly post the time when the problem occured. The publish time of the last post could be used to calculate the time span of a thread, which is used for indexing methods (§2.2.2). The bodies of the posts are used for keyword extration (§2.3).

To extract the features from NANOG forum, we first download all posts in HTML format from NANOG forum. We implement an algorithm that recursively cleans up nested quotations in every downloaded post. We reason that quotes are not actual contents of the posts, but instead repetition from previous contents. They would falsely increase the importance (i.e., appearances) of the quoted words, and affect the weight of words on the word frequency matrix. After the removal of quotes, we implement a detagger to convert the posts to plain text format. We choose not to parse the post in HTML format directly because there is no any tags inside the message body that would help distinguish the semantic of various parts in a post. Based on our observation, the body of a post, including the signature and public key digest, are enclosed in a single `<pre> </pre>` pair. Therefore, to work with the posts in plain text format, the detagger is implemented to elimiate the noises caused by HTML tages.

Besides, while plain-text posts have better readability, there remains some noisy words. For example, the phrase "North American Network Operators Group" appears in every post as header and structural words. We implement a simple word-processing program to automatically filter out these structural contents. However, although our word-processing program can easily filter out the noises that appears *repeatedly*, it remains another type of noises. Specifically, the message contents also include the signatures, message digest, manual quotes that are not removed by HTML detagging and other noises. Filtering out the signatures poses a tough issue because a lot of signature include email addresses, company titles and internet addresses, which are parts of the information we try to retrieve from the contents. Filtering out these words therefore would be critical for the performance of the name entity recognition (§2.2.1). After observing the common pattern of the signature, we deploy a heuristic that, starting from the end of a post, we discard lines of the thread until we reach the first line that ends with a period, question mark, exclamation mark or until it reach a line that starts with two dashes.

After we clean up the single post, we form a thread by combining the posts with the same title. We order the bodies of posts in a thread by its publish time.

2.2 Classification

In this stage, we aim to classify the discussion units into outages categories. Previous work by Feamster and Balakrishnan [12] analyze the forum manually and classified them into 10 categories. We started off to adopt their categories and try to manually label the forum posts since year 2007 to October in year 2009. However, we observe that most of the categories in Feamster and Balakrishnan's paper is very granular. For example, the most common category, global route visibility, has only 146 threads discussing actual network problems. By our manual inspection, we identify about 150 threads that are discussing actual network outages during year 2007 and 2008, while the total number of threads that are over 2000. The number of threads does not justify a granular classification from the beginning. Therefore, we adopt a two level filtering and classification approach. We first classify threads into either network problem related or not-related, then further classify them in sub-categories. The performance of this scheme is discussed in Section 3.

The output from the first stage, i.e the combined contents of a thread, is then tokenized, processed and filtered by a chain of recognizers (§2.2.1), and converted into a term occurrence vs. document matrix.

The matrix is then undergone indexing to transform into a weighted term frequency vs. document matrix (§2.2.2). We combine this matrix with manually selected features, and form a final feature vs. document matrix (§2.2.3). We uses the classification methods implemented in Weka Machine Learning library [14] to classify our documents. Several popular algorithms for text classification such as Naive Bayes, SVM and decision tree are used and evaluated (§2.2.4).

2.2.1 Document Processing

We construct a program tokenizer using the rule-based break iterator from International Components for Unicode package [3]. There were other available packages available for us to choose from, specifically Java's inherent text package and the text package from Apache [1]. We choose to use the ICU package because it provides users a convenient user-defined rules using regular expressions. We extend their rule-based tokenizer with IP addresses (e.g., 192.168.2.1), IP prefixes (e.g., 192.168.0.0/16), and general Internet site addresses (e.g., `http://www.cs.illinois.edu/homes/pbg/`), and email addresses. IP prefixes have higher priority than IP addresses. The general Internet site addresses is used to implement a heuristic in §2.2.3.

The reason we need to recognize IP prefixes, IP addresses and Internet site addresses are motivated by the observation that people reporting a network outages would post websites or IP addresses they can not reach, or domains that they find out having problems. The responding posts would often include traceroute traces or other network tracing software reports. These are all valuable information that we deem important to extract. Despite the fact that these traces come in various format, they include the IP addresses or domains that the traces went through and potential problematic sites.

We then pass the tokens throw a chain of recognizers. The recognizer is a Java interface we implemented for flexible incorporation of different token recognition algorithms. In this paper, we implemented two token recognizers. The first recognizer is a stop word recognizer. We used the standard stop-word list retrieve from Onix Text Retrieval Toolkit [5]. The second recognizer is a general name entity recognizer from Stanford University NLP Group [6]. The name entity recognizer can recognize token type date, organization, location and human name. The recognizer is a general purpose one, trained on the CoNLL 2003 [2] English training data. We reason that a general English recognizer would suffice for our task because we mainly use the name entity recognizer to recognize organization names, such as AT&T or Comcast, and locations where outages occurs. These two categories are both recognized by the general English recognizer.

2.2.2 Term Frequency Indexing

Term indexing, or weighting, is a common practice for changing the weights of a term frequency to better reflect its influence on indicating the category of a document. Various indexing approaches had been proposed [22]. In this paper, we consider five widely-used indexing schemes:

- **Term Frequency (TF)**, which sets the weight for word w in document D to the number of times w appeared in D.
- **Document Frequency (DF)**, which sets the weight for word w to the number of documents in the corpora that contain this word.
- **Inverse Document Frequency (IDF)**, which sets the weight for word w to the inverse of its Document Frequency.
- **TFIDF Indexing**, which sets the weight to the product of its term frequency and its inverse document frequency. In our implementation, we uses a logarithmic inverse document frequency with Laplace smoothing.

Mathematically it is expressed as:

$$a_{ik} = f_{ik} \times \log\left(1 + \frac{N}{n_i}\right)$$

where a_{ik} is the weight of word i in document k, f_{ik} represents the document frequency, and the log term is the Laplace smoothed inverse document frequency.

- **Latent Semantic Indexing**, which is an indexing algorithm that measure how closely related two documents are.

The intuition for TF indexing is that, if a word appears frequently in a document, it should be more representative of its actual category. It suffers from a problem that it gives too much weight to common words such as "internet" that are typical in network related issues. Word document frequency indexing is not commonly used for classification problems, but instead is used in finding hierarchy of documents [17]. IDF would give a word that's rare across the corpora a higher weight. The intuition is that, is a word is more rare across the corpora, it would be more indicative about the category of this document. It suffers from a problem that, it is too sensitive to an accidental occurrence of a word in a document. Such accidental occurrence can happen due to a word having different meanings. The TFIDF indexing combines the advantages of both indexing methods while mitigating their problems. Latent Semantic Indexing [10] is a mathematically methods that try to reduces the dimension of the matrix and discover underlying relations between features.

2.2.3 Features Weighting for Classification

While the basic *term-vector* algorithm behavea well in general cases, specific features could enhance the performance of classification program for the corpora at hand. We present a general framework to incorperate the human observations into classification by weighting features. Observed features would be assigned a higher weight to distinguish it from pure statistical weight, thus having higher indicating effect during classification.

We first observe that, for threads reporting an outages, the initial first few replies come in within hours. These replies contains most useful information such as traceroutes to the reported website, news about the mentioned organization, etc. Replies that come later usually digress from the actual topics and discuss other topics. Therefore, more weights should be given to terms that occur in the responses posted closer to the initial post dates while less weights should be given to terms occur later in the reply. Therefore, we design an exponential discounted scheme to weight the terms in a series of replies. Specifically, we discount each term by

$$e^{-(n/a)} \tag{1}$$

, where n is the number of days document in which word appears is posted after the main thread. As a result, replies posted during the same day as the initial post would not be discounted, while replies posted later would get their weight reduced. We aimed to discounted the weight such that the term for threads posted after 3 days the initial thread is posted would be discounted by a half. Because $ln(0.5) = -0.7$, we reason that by choosing $a = 4$, we would obtain a good approximation for our target.

In additional to the above observation, we also make the following observations.

- Presence of general Internet site addresses in the first thread.
- Length of the first post.
- Whether the first post contains the following keywords: anyone, anybody, everyone or everybody.

The first heuristic is from the observation that, for discussion purpose threads, authors frequently posted a news link for background of the discussion. On the contrary, people discussing a outages would not post a general website, but instead the domain of the website. For example, when discussing about CNN outages, author would post the link cnn.com instead of a specific news article address from cnn.com. Therefore, we reason that presence of general website addresses in the first post would be a indication of a discussion thread.

The second heuristic shares similar intuition as the first one. We observe that if people are trying to discuss a topic, in addition to provide background new articles, they also include their opinions, which in most cases are long in length. For threads discussing network problems, however, the first thread are usually short in their length. The author usually give succinct description about the network problem and ask for help. Therefore, we reason that a long thread would indicate a discussion thread instead of a problem reporting threads.

The last heuristic comes from how people usually ask for help or solicit information from other operators on the mailing list about the network problem. The person trying to report a problem would ask whether other people experience similar problem, or provide traces for diagnosis purpose. Such request are usually written in sentences with the mentioned keywords. Such feature has already been captured by the TFIDF indexing, but we reason that it is important enough to justify a stand-alone feature.

2.2.4 Classification Algorithm

A lot of classification algorithms have been shown to perform well for general text classification problems. We use three classic algorithms in our evaluation. The algorithms are discussed below.

Naive Bayes

Naive Bayes is the classic model for document classification. It models a document as a bag of words. Each word has certain probability appearing in a document in a certain category. The model also assumes the probability that a word occurs in a document is independent. The probability a word occurs is estimated as an maximum expectation estimation, i.e, dividing the number it occurs in the training corpora by the total number of words in the corpora. The assumption that words occurring in a document independently is not true in general case, but has been shown to perform surprisingly good in a lot of tasks, including the binary classification tasks that we are considering here.

Decision Tree

In this approach, a tree is constructed from the training data. Each node in the tree represents a feature that determines which branch a given document would follow. A new instance of document to be classified will start from the root node and traverse along the edges until it reaches a tree node, in which a category is assigned. Decision tree learning algorithm performs better when the number of features are small, and some dominating features exist that are would strongly indicates the type of a document. The algorithm used to construct the tree varies, and technical details are beyond the scope of this paper. In our work, we adopt the common C4.5 decision tree algorithm [19].

Support Vector Machine

Support Vector Machine with linear kernel has been considered as one of the best, if not the best, off the shell general classification algorithms available [18]. SVM models each documents as a vector in a high dimensional vector spaces, and it try to identify a linear function in this vector spaces that would linearly separates the two categories while maintaining the largest possible margin to the two separated clusters. The vectors from the two clusters that are closest to the linear plane is called the support vectors for the algorithm. The SVM algorithm has advantages in that it has a computationally efficient implementation called SMO that can handle the high dimension input spaces efficiently. In general, most text categorization problems are linearly separable [22], and the input space is extremely high dimension due to fact that a corpora usually has a large vocabulary. SVM seems to suite the task of classification well.

2.3 Keyword extraction

Much of the Keyword extraction tasks has been described in previous sections when we are generating classification features. Here, we notice that, logically, keyword extraction is a separate part from the classification task in that it is trying to mind useful information from the threads. The information minded may or may not be useful for classification. A feature that is common to both type of threads may have little value for the task of classification, but it can nevertheless be useful information, such as domain names and network corporation names. During the initial design phase, we planned to separate the thread filtration and classification part and the keyword extraction part into two physically separate parts of the project. However, the two parts merges under the same framework during the implementation phase as a step in generating the feature matrix.

3. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our proposed algorithm. In Section 3.1, we analyze the accuracy of our text cleaning heuristics. In Section **??**, we analysis the performance of our indexer and classifier in terms of recall, false positive rate and false negative rate. Then we report the results of our keyword extraction mechanisms on threads identified as reporting problems in Section **??**. We evaluate the usefulness of the keywords by correlating them to routeview data in Section 3.3.

3.1 Text Cleaning

Category	fraction
Satisfying	80.4%
Over-pruning	7.8%
Under-pruning	11.8%

Table 1: Performance of filtering heuristics

We use some heuristics in the format of the messages to filter out noise contents from the plain texts (as discussed in §2.1). We manually look at 51 threads pick randomly from threads with such noises and the subsequent threads generated and evaluate the results. As shown in Table 1, we put the results into three categories. First category is satisfying, which means noise words are being filtered out without affecting any essentially content words. Threads are classified as Over-pruning if some content words are mistakenly parse out. Threads are classified as Under-pruning if there remains some noises. The fractions of the threads in each category are summerized in Table 1, where above 80% threads are filtered correctly using our heuristics.

3.2 Classification

Let $tp(tn)$ and $fp(fn)$ respectively be the true positives (negatives) and false positives (negatives) produced by classifier. Then the metrics for evaluating the performances of algorithms that classifies outages are:

- Precision, which is the number of true positive instances divided by the number of positive instances classified. It measures how many instances retrieved are relevant to the topic. Mathematically, it is expressed as:
$$\frac{tp}{tp+fp}$$
- Recall, which is the number of true positive instances divided by the number of actual positive instances. It measures how many relevant items are retrieved, i.e, the sensitivity of the algorithm. Higher recall means the algorithm is more sensitive. It is expressed, mathematically, as:
$$\frac{tp}{tp+fn}$$
- False Positive Rate, which is the number of false positive instances divided by the actually number of negative instances. It measures how specific the classification algorithm is. The higher positive rate, the less specific it is in classification. Mathematically, it is expressed as:
$$\frac{fp}{fp+tn}$$

For our task, we focus on optimizing the recall instead of improving the precision. The reason is that while high recall can result in a low precision by having retrieve too many false positive instances, our goal is to extract information from the threads that can be utilized to diagnosis potential network problems. Therefore, missing threads for network problem discussion (i.e, false negatives) is more severe than retrieving a thread that is not discussing about network outages (i.e, false positives). In short, we trade sensitivity for specificity.

We first evaluate the baseline document vector model classification, which means we use only words in the document as features without adding in our manually selected features. For indexing, we implemented word frequency indexing, inverse document frequency indexing, TFIDF indexing, latent semantic indexing and exponential discounted indexing. For classification, we uses the classification algorithms in the Weka Machine Learning library. Specifically, we uses their implementation of Naive Bayes, C4.5 Decision Tree and SMO algorithms.

We first run out program on all the available data. However, we quickly realize that the computing resources we have, which is a Macbook Pro with 2GB memory and 2.16GHz Intel Core 2 Duo running Snow Leopard, is not able to handle such a work load as it constantly run into java heap space problem when we are trying to construct the raw term document matrix. We therefore reduces the size of our corpora to include data from the first 6 months of year 2007, which contains 30 threads discussing network outages and 245 other threads.

During the indexing process, we were not able to run the latent semantic indexing method again due to memory and java heap spaces issues. We run the program on a corpora with 10 documents and they were successful. We suspect that the matrix decomposition operations used in LSI, as implemented by the generic apache matrix library we used, is computationally expensive and does not scale well to corpora with over 100 documents. We would like to revisit this indexing scheme in the future when we get access to better computation resources.

The performance results are presented in Table 2. We subsequently incorporate the manually selected features into our feature spaces, apply exponential discounted weighting on the already existing term weights, and further prune the feature spaces to include terms happening only more than 3 times to reduce noises. The performances are summarized in the Table 3.

We focus on comparing the performance differnces between the baseline term vector model algorithm and our optimized algorithm. We observes that our optimization improves the precision and fp rate of classification. With optimization, we also observe that the recall of SMO increases by 3% to 7%.

Comparing the three algorithm, we observe that Naive Bayes has generally highest FP rate, indicating that Naiye Bayes has property of high sensitivity but low selectivity. SMO generally have the lowest FP rate. This shows that SMO has higher selectivity than sensitivity.

3.3 Correlation to network logs

In this section, we conduct two experiments to evaluate the performance of our proposed method by correlating the extracted keyword to real network logs. For each experiment, we randomly select an outage reported by our method. To validate that the extracted information are useful for detecting network anomaly, we replay the network logs (BGP RIBs and updates) collected by six Zebra routers from Route Views project [7].

Algorithm		Naive Bayes	C4.5 Decision Tree	SMO
TF	Precisoin %	46.8	66.7	63.0
	Recall %	68.6	76.2	64.8
	FP Rate %	33.5	16.3	16.3
IDF	Precision %	38.7	59.0	71.4
	Recall %	89.5	59.0	47.6
	FP Rate %	60.8	17.0	8.2
TFIDF	Precision %	50.3	66.7	68.1
	Recall %	73.3	74.3	61.0
	FP Rate %	31.0	15.9	12.2

Table 2: Performance with baseline document vector model

Algorithm		Naive Bayes	C4.5 Decision Tree	SMO
TF	Precisoin %	52.2	72.0	73.1
	Recall %	79.0	63.8	72.4
	FP Rate %	31.0	10.6	11.4
IDF	Precision %	39.7	63.8	79.3
	Recall %	84.8	72.0	65.7
	FP Rate %	55.1	10.6	7.3
TFIDF	Precision %	50.3	76.6	67.7
	Recall %	76.2	68.6	63.8
	FP Rate %	32.2	9.0	13.1

Table 3: Performance with optimization

In the first experiment, the reported outage is a discussion unit with thread title "Anomalies with AS13214 ?" posted on July 28th, 2009. The extracted information from this outage is the AS number "13214". Figire 2 shows the number of received update messages (i.e., announcement and withdraw messages) and of announced prefixes originated from the AS 13214 from the Zebra routers on July 28th, 2009. Notice that the bilateral arrow in the Figure 2 represents the time interval between the time of the first post and the last post from the discussion unit. We observe that the Zebra servers received thousands of BGP announcements originated from AS 13214 in only one hour period (8 to 9 AM), while the first post discussing the problem appeared around 10 AM.

In the second experiment, we analysis an outage reported on Jan. 4th, 2007. Instead of reporting any suspicious AS, our algorithm reported a prefix related to the outage. Again, we leverage network logs stored at Zebra routers of the Route View project. In this experiments, we evaluate the reachability of any reported prefix. The reachability of a prefix is defined by the fraction of the Zebra routers that have at least one valid path to reach the prefix. For the prefix `194.153.114.0/24`, Figure 3 shows that the prefix is not reachable until 12:50, Jan. 5th, 2007, which is about 24 hours after the time of first post of the discussion unit.

Besides to the above two cases, the following problems are also validated from Zebra routers:

prefix instability: Using the longest prefix matching algorithm, we measured that some reported IP changes their using prefix frequently. It happens when the router receives a withdraw of less-specific prefix or an announcement of more-specific prefix for an IP address.

AS-level path instability: We validated that some reported IP are flapping (i.e., switching the forwarding path frequently) during the reported time.

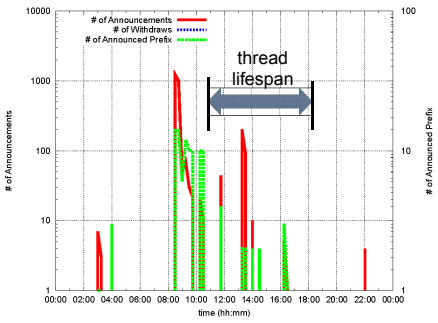

Figure 2: The number of received updates/prefixes originated by AS 13214 on Jul. 28th, 2009.

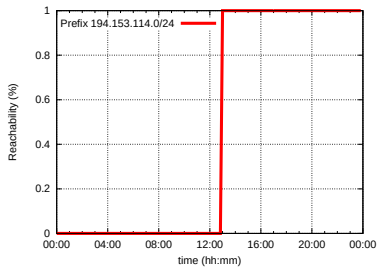

Figure 3: The reachability of prefix 194.153.114.0/24

4. CONCLUSIONS

This paper proposes a new method to aid anomaly detection by extracting information from NANOG forum. We adopt novel machine learning techniques such as SVM and NER to efficiently obtain the network outage information. With the proposed methid, the network operators can focus on tracking suspicious ASs, IP prefixes, or domain names reported by our method with high detection probability. We believe our method can cooperate with other anomaly detection/prevention algorithm towards collaborative troubleshooting.

The main problem we confront is that the forum threads are mostly short in its nature and heterogeneous in its features. For future research, we would focus on using more advanced classification model than the common term vector model, and deeper semantic analysis to extract or infer more relavant features from forum posts.

5. REFERENCES

[1] The apache software foundation., `http://www.apache.org`.

[2] Conll: the conference of acl's special interest group on natural language learning.,`http://ifarm.nl/signll/conll/`.

[3] Icu - international components for unicode.,`http://site.icu-project.org/`.

[4] The north american network operators group (nanog) mailing list archive, `http://www.cctec.com/maillists/nanog/`.

[5] Onix text retrieval toolkit, `http://truereader.com/manuals/onix/stopwords1.html`.

[6] Stanford named entity recognizer.,`http://nlp.stanford.edu/software/CRF-NER.shtml`.

[7] University of oregon routeviews project., `http://www.routeviews.org/`.

[8] H. Ballani, P. Francis, and X. Zhang. A study of prefix hijacking and interception in the internet. In *SIGCOMM'07*, pages 265–276, New York, NY, USA, 2007. ACM.

[9] H. Ballani, P. Francis, and X. Zhang. A study of prefix hijacking and interception in the internet. *SIGCOMM Comput. Commun. Rev.*, 37(4):265–276, 2007.

[10] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[11] N. Feamster and H. Balakrishnan. Detecting bgp configuration faults with static analysis. In *Networked Systems Design and Implementation (NSDI) 2005*.

[12] N. Feamster and H. Balakrishnan. Detecting bgp configuration faults with static analysis. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 43–56, Berkeley, CA, USA, 2005. USENIX Association.

[13] S. Goldberg, S. Halevi, A. D. Jaggard, V. Ramachandran, and R. N. Wright. Rationality and traffic attraction: incentives for honest path announcements in bgp. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 267–278, New York, NY, USA, 2008. ACM.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[15] J. Karlin, S. Forrest, and J. Rexford. Autonomous security for autonomous systems. *Computer Networks*, 52(15), 2008.

[16] R. Mahajan, D. Wetherall, and T. Anderson. Understanding bgp misconfiguration. In *ACM SIGCOMM 2002*.

[17] A. Medem. Troubleminer: Mining network trouble tickets.,`www-rp.lip6.fr/~medem/im_workshop_09_amelie_medem.pdf`.

[18] A. Ng. Stanford machine learning course nodes.,`http://www.stanford.edu/class/cs229/materials.html`.

[19] J. R. Quinlan. *C4.5: Programming for machine learning*. Morgan Kaumann, 1993.

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

[21] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz. Listen and whisper: security mechanisms for bgp. In *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[22] F. I. Thorsten Joachims and L. Viii. Text categorization with support vector machines: Learning with many relevant features, 1997.