SMU Data Science Review

Volume 2 | Number 1 Article 5

Border Gateway Protocol Anomaly Detection Using Machine Learning Techniques

Philip Edwards
Southern Methodist University, pbedwards@smu.edu

Lu Cheng
Southern Methodist University, lucheng@smu.edu

Girish Kadam
Ericsson, eusghkm@yahoo.com

Border Gateway Protocol Anomaly Detection Using Machine Learning Techniques

Lu Cheng[1], Phil Edwards[1], Girish Kadam[2]
[1] Master of Science in Data Science, Southern Methodist University,
Dallas, TX 75275 USA
[2] Ericsson,
6300 Legacy Dr, Plano, TX
{lucheng, pbedwards}@smu.edu
{eusghkm}@yahoo.com

Abstract. As the primary protocol used to exchange routing information between network domains, Border Gateway Protocol (BGP) plays a central role in the functioning of the Internet. Border Gateway Protocol is a standardized router protocol used to initiate and maintain communication between domains, or autonomous systems, on the Internet. This protocol can exhibit anomalous behavior caused by improper provisioning, malicious attacks, traffic or equipment failure, and network operator error. At large internet service providers, many BGP issues are not immediately seen or explicitly monitored by network operations centers. This possible blind spot is due to the enormous number of BGP handshakes that occur throughout the network along with the fact that there are many of these sub-interfaces associated to a single physical connection. We will present machine learning methods for anomaly detection using unsupervised learning techniques and discuss possible data pipeline methods to quickly collect and trigger on these anomalies when they occur. Clustering techniques including *k*-means and DBSCAN were successfully implemented and able to detect known anomalies for historical events. This approach could incur soft savings by triggering early detection warnings of anomalous BGP events, but human intervention may still be required in order to address possible false positives.

1 Introduction

At large Internet Service Provider (ISP) networks, Border Gateway Protocol (BGP) is the most widely used exterior gateway routing protocol for domain peering. When establishing these peering connections, BGP exchanges routing and reachability information between domains or autonomous systems (AS) on the Internet. The handshake to initiate a BGP session is a finite state machine that transitions between six different states. The patterns to get from ‘idle’ to ‘established’ are not always the same and even repeat over and over in some cases. The timestamp, current and previous state, and other information related to BGP state changes are sent to router log servers as messages where they can be collected and analyzed. These interactions are reported as BGP neighbors periodically drop and reestablish, which makes this data available for network alarming perspective. [1]

Some examples of historical anomalies that affected BGP routing performance are listed below. Some of these were selected for to apply unsupervised learning to as they were massive scale events and easier to measure results. This project would attempt early detection of similar type events as some of these events reoccurred or continued to create havoc for many days as many of these were not detected and/or categorized in a timely manner.

- TTNet announced more than 100,000 incorrect routes, 24 December 2004
- TMNet BGP misconfiguration, 12 June 2015
- AS27506 hijacked panix domain, 14 January 2005
- Dodo ISP incident, 23 February 2012
- Mosco blackout hardware failure, 7 May 2005
- Worm attacks such as Slammer, Nimda, Code Red I

The motivation for this project is to introduce soft savings associated with early detection and isolation of BGP events.

- BGP can sometimes exhibit anomalous behavior caused by improper provisioning, malicious attacks, traffic or equipment failure, and network operator error.
- At large service providers, many BGP issues are overlooked by network operations centers (NOC) because they are a logical sub-interface of a larger physical interface that stays intact (critical alarming occurs only on the physical interface).
- BGP handshakes are constantly occurring so they are mostly ignored as normal operation based on high volume vs. low actual event incidence.

It is possible to make use of BGP message logs as they are collected by applying machine learning techniques to detect anomalous behavior that could be network affecting. During this effort, false positives must be kept to a minimum as it is extremely important to not overload already busy operations center with useless information. [2]

The simple problem definition is to find BGP anomalies and report them. The approach for this paper is to address the problem by implementing unsupervised clustering techniques to detect anomalies and iteratively apply them to datasets of multiple known events as a metric to measure against to determine optimal model(s) and parameterization. Once the optimal clustering method or combination thereof is determined along with parameterization, the next step is to implement a near real-time data flow to apply to continuous data collection.

The process flow for model selection and parameterization is a static process based on historical data, and a near real-time data pipeline would effectively use the selected model and its parameters to constantly monitor BGP messages as they are collected. Some of the process step highlights are as follows:

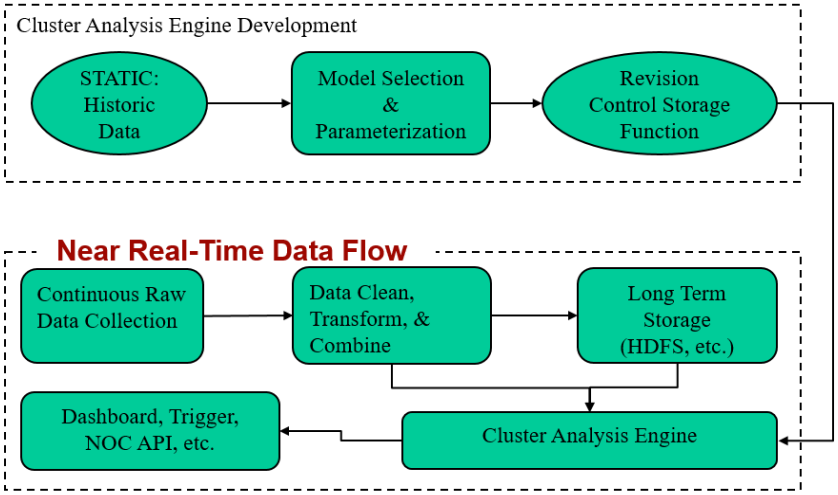


Fig. 1: Process Flow Diagram

The upper box contains offline model selection and parameterization process. Once the optimal model(s) are selected and parameterized, they are testing by applying to historical known events. The final step is to apply model(s) to near real-time data collected every *n* minutes via scripted batch process.

The flow of the upcoming sections of this document will begin with an informative BGP primer followed by related work examples. After that, the document will step through some of the process steps in Figure 1 as shown on the list below.

1. Data collection and preparation.
2. Model selection, parameterization, and testing.
    a. Results vs. historical event data.
    b. Different clustering techniques.
3. Near real-time data flow pipelining overview.
4. Conclusions.

These steps imply the implementation of a data pipeline that collects, cleans, analyzes, and then triggers based on offline model creation applied to near real-time BGP data. Reproducible research during initial model creation (first three steps above) is essential to ensure production pipeline capability.

Five historical events were selected, and clustering techniques including *k*-means and DBSCAN were successfully implemented and able to detect known anomalies for these events. False positives are still an issue during known historical 'quiet times', however, which would require human intervention. This approach could still incur soft savings (customer uptime satisfaction, labor reduction, latency, etc.), by sending a report of BGP anomaly event probabilities but is not yet capable of detecting anomalies without any false positives.

2 BGP Primer

BGP has a long history that goes back to early 1980s. It was developed to help interconnected gateways to efficiently exchange routing information. These gateways connect different networks, and these networks are independently managed by their own administrative authorities, and they are called Autonomous System (AS). A typical enterprise network can have one or more AS numbers. Within an internetwork, such as an enterprise network, routers typically use Interior Routing Protocols. These routers are under the same administrative authority. Beyond these networks and at the border gateways, Border Gateway Protocol (BGP) is used. An enterprise network connects upstream to a provider network, and the provider connects to high level providers, and now this becomes a network of networks. **[3]**

BGP was introduced as a truly reliable dynamic routing protocol for inter-AS routing. It is classless and supports Classless Inter-Domain Routing (CIDR). It relies on TCP (port 179) for maintaining the neighboring relations with peer border gateway routers. The TCP mechanism efficiently handles activities such as handshakes, acknowledgement, retransmission and sequencing. Each pair of peers maintain a point-to-point session. **[3]**

BGP is a vector protocol like RIP, and Each BGP node obtains routes from their downstream neighbors. It then processes and calculates its own routes. The results are then advertised to upstream neighbors. Its calculation bases on a path vector, and one of the path attributes is AS_PATH. The calculation is to find the shortest inter-AS path to reach the destination.

BGP is a loop-free routing protocol. Route loops can be easily detected using AS_PATH attribute.

In summary, BGP (version 4) is an interdomain routing protocol designed to provide loop-free routing links between organizations. RFC1771 introduced and discussed several new BGP features to allow the protocol to scale for internet use. Most Internet Service Providers use BGP as their border routers standard routing protocol. Organizations use BGP to connect to an external network such as provider network to gain access to the Internet.

There are some unique characteristics about BGP:

- BGP peers use a point-to-point unicast connection
- BGP is an application layer protocol using TCP (port 179). Session maintenance comes from TCP functions such as acknowledgement, retransmission and sequencing
- BGP is a path vector protocol using autonomous systems numbers
- BGP routes uses a route attribute called AS_PATH, and list AS numbers in sequential set
- The shortest path in the AS_PATH attribute determines the best path to the destination
- The AS numbers on the AS_PATH helps detect any loop

2.1 BGP Messages

BGP has the four basic message types shown in Table 1.

Table 1. BGP Messages

Type Name	Function Overview
OPEN	Sets up and establishes BGP adjacency
KEEPALIVE	Ensures that BGP neighbors are still operating
UPDATE	Sends routing updates to peers
NOTIFICATION	Indicates an error condition to a BGP neighbor

Due to data availability, this project will mainly focus on analyzing BGP update messages. Peers use update messages to synchronize routing tables. When one peer router has changes in its routing table, it will send an update message to inform the other router peer. The objective of using this message is to let other networks know about these network changes. Update message contains the feasible routes (announced), withdrawn routes or both. It also includes Network Layer Reachability Information (NLRI) and path attributes. Each update message only describes a single BGP path.

2.2 Path Attributes

A path attribute is a characteristic of an advertised BGP route and is included in the update message. It contains information about the destination such as next hop address. Path attributes are essential in BGP route calculation. Attribute usage of well-known mandatory indicates the attribute must be included in the BGP message or the session is closed (with a notification error generated). The optional attributes are passed along to other peers if transitive and simply ignored if nontransitive.

Table 2. Path Attributes

Attribute Name	Attribute Usage
ORIGIN	Well-known mandatory
AS_PATH	Well-known mandatory
NEXT_HOP	Well-known mandatory
LOCAL_PREF	Well-known mandatory
ATOMIC_AGGREGATE	Well-known mandatory
AGGREGATOR	Optional transitive
COMMUNITIES	Optional transitive
EXTENDED COMMUNITY	Optional transitive
MULTI_EXIT_DISC (MED)	Optional nontransitive
ORIFINATOR_ID	Optional nontransitive
CLUSTER_LIST	Optional nontransitive

AS4_PATH	Optional transitive
AS4_AGGREGATOR	Optional transitive
Multiprotocol Reachable NLRI	Optional transitive
Multiprotocol Unreachable NLRI	Optional transitive

Optional transitive – If the attribute is not recognized by the BGP implementation but the transitive flag is set the attribute should be accepted and passed along to other peers.
Optional non-transitive – If the attribute is not recognized by the BGP implementation but the transitive flag is not set the attribute should be ignored and not passed on to other peers.

2.3 BGP Session Establishment

There are two phases of BGP session establishment: TCP connection establishment phase and BGP session establishment phase. TCP connection must be established for BGP. This is called three-way handshake. The session is initially in Idle state. With TCP connection established, the session state changes to connect. After that, BGP enters OpenSent, OpenConfirm and Established states.[4][5]

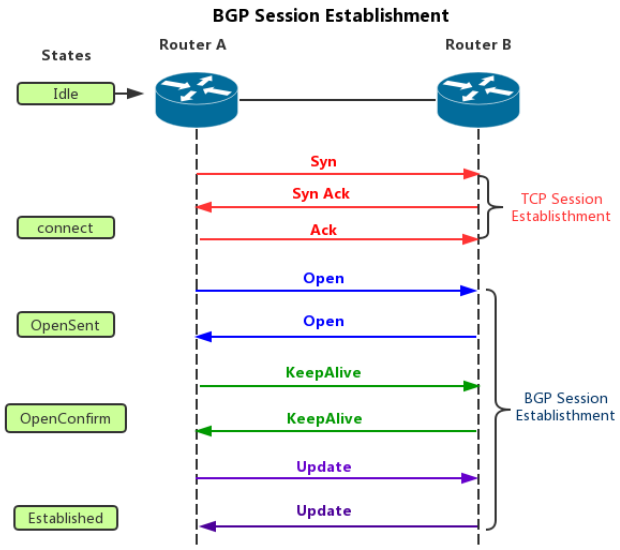


Fig. 2. BGP Session Establishment [5]

3 Related Work

Security is a critical requirement when organizations connect their network to the Internet. However, BGP has many well-known security vulnerabilities. It is a protocol that multiple independent organizations use to exchange routes in the public internetwork. These organizations typically belong to different management domains or Autonomous Systems. When there are security vulnerabilities, the malicious actors will try to exploit for their benefits.

Some of vulnerabilities are inherent to the basic BGP architecture and framework. The BGP protocol does not require strict integrity check and it does not enforce strict authentication of the handshake messages. Most of the update message does not need to authorize the senders.

Malicious actors can purposely inject bad routes via crafted BGP messages to cause damage to the organization outbound traffic. Peers exchange prefixes with the best routes, and they are locally defined. The BGP route selection process and algorithm lack the proper security mechanism to detect and reject bad routes that sent from a bad neighbor.

BGP works when the peering gateway routers operate in the correct operation model. When the operation is disrupted by a successful attack, the routing process can be degraded. Routers can be overwhelmed with routing messages. CPU and Memory resources can be exhausted. The correct operation depends on the integrity, authenticity and timeliness of the routing information BGP peers send. BGP routers are required to process, store and distribute this information in accordance with both the BGP specification and local routing policies.

Detecting a security issue is difficult. There are many limitations to the ability of any practical security mechanism to monitor all BGP messages. Any external observer cannot easily determine if a neighboring router is operating BGP in the proper way. This is because monitoring such behaviors of a neighbor is beyond its local management or local AS. One example of the security issues is that BGP does not require sequence numbers. A bad router can send an UPDATE based on authentic but outdated information. **[6]**

Some of the security issues can be un-intentional by a malfunctioning router or a bad configuration. A router advertising the wrong route to a prefix can damage or blackhole another organization network traffic. There are many incidents reported in the past about Internet meltdowns resulting in widespread loss of use for several days in some cases.

The challenge is big, and the industry has been looking for effective solutions. There are many ongoing researches to secure BGP. BGP events can be collected and analyzed. BGP behavior can be observed, and good and normal behaviors are baselined. Abnormal behaviors can be detected, and the bad traffic can be rejected to stop causing more damages to the Internet.

National Institute of Standards and Technology (NIST) has published a set of security guidelines and recommendations to help organization to secure BGP. Access list Control, peer authentication, and prefix limits are some of the methods that have been recommended. To be more proactive in finding the threats, researchers are looking for more effective way to identify bad behavior before any harm has been done. **[7]** NIST provides a guidance for the evaluation of BGP anomaly detection.

Researchers have been using different techniques to detect any abnormal BGP behavior. They baseline the normal pattern and identify signatures of anomalies. Wavelet analysis is one of the techniques that can reveal the time-frequency behavior. The researchers take advantage of the strength of wavelet analysis in handling signal with scaling property and its ability to detect network anomaly and identify network wide anomalous events. Their method can detect network-wide events such as message volume surges caused by slammer worm attack [8].

Other researchers are studying the hierarchy of abnormal BGP Events. They want to classify the abnormal BGP events based on a hierarchy discovered by clustering method. They apply data mining techniques to build a tree-like hierarchy of abnormal BGP event classes [9].

4 Dataset

The source data for this project will be collected from the Réseaux IP Européens Network Coordination Centre (RIPE NCC) which stores public Internet routing data through the Routing Information Service (RIS). This data is made available for researchers without restrictions. This routing data is collected from approximately two dozen (exact count varies over time) Remote Route Collectors (RRC) around the globe. Prior to July of 2003, BGP messages were collected and stored at fifteen-minute interval with the message sampling rate increased to every five minutes since that date. **[10][11]**

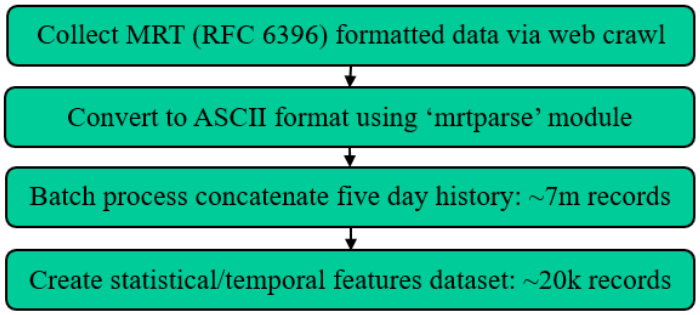


Fig. 3. Data Collection and Parsing Steps

The collected BGP update messages are stored in multi-threaded routing toolkit (MRT) format, which is described in RFC6396. The MRT formatted BGP update messages are converted into ASCII format by slightly modifying a module found on Github called mrtparse. This module is used as a part of the preprocessing steps as follows. First, the RIS raw data is collected via web crawl script. The next stage includes an iterative bash script to process all collected data files through mrtparse module, and the parsed outputs collected in batches. Finally, concatenation of all the message batches is performed in Python at which point the data is finally prepared for

exploratory data analysis (EDA), feature engineering, etc. The BGP remote route collector used for this project is RRC04 deployed at CERN Internet Exchange Point (CIXP), Geneva, Switzerland. Dependent variable for supervised learning will be based on known periods of Internet anomalies. **[12] [13]**

An example record of the parsed output is shown in Figure 4.

```
MRT Header
    Timestamp: 1543622399(2018-11-30 17:59:59)
    Type: 16(BGP4MP)
    Subtype: 4(BGP4MP_MESSAGE_AS4)
    Length: 149
BGP4MP_MESSAGE_AS4
    Peer AS Number: 6939
    Local AS Number: 12654
    Interface Index: 0
    Address Family: 2(IPv6)
    Peer IP Address: 2001:478:124::176
    Local IP Address: 2001:478:124::171
BGP Message
    Marker: -- ignored --
    Length: 105
    Type: 2(UPDATE)
    Withdrawn Routes Length: 0
    Total Path Attribute Length: 82
    Path Attribute Flags/Type/Length: 0x40/1/1
        ORIGIN: 0(IGP)
    Path Attribute Flags/Type/Length: 0x40/2/14
        AS_PATH
            Path Segment Type: 2(AS_SEQUENCE)
            Path Segment Length: 3
            Path Segment Value: 6939 7473 4804
    Path Attribute Flags/Type/Length: 0x40/6/0
        ATOMIC_AGGREGATE
    Path Attribute Flags/Type/Length: 0xc0/7/8
        AGGREGATOR: 65367 10.194.27.34
    Path Attribute Flags/Type/Length: 0x80/14/44
        MP_REACH_NLRI
            AFI: 2(IPv6)
            SAFI: 1(UNICAST)
            Length: 32
            Next-Hop: 2001:478:124::176 fe80::ce4e:24ff:fe94:3f30
            NLRI: 2405:dc00:35c::/48
```

Fig. 4. Parsed MRT Record Example

4.1 Exploratory Data Analysis (EDA)

The individual parsed messages as shown in Figure 3 are converted to long format, filtered to only include BGP UPDATE messages which leads to the following output fields.

For withdrawn routes, the pipe delimited fields are:

BGP protocol | ***timestamp*** | ***Withdraw or Announce*** | ***PeerIP*** | ***PeerAS*** | ***Prefix***

For announcements, the pipe delimited fields are:

BGP protocol | ***timestamp*** | ***Withdraw or Announce*** | ***PeerIP*** | ***PeerAS*** | ***Prefix*** | ***AS_PATH*** | ***Origin*** | ***Next_Hop*** | ***Local_Pref*** | ***MED*** | ***Community*** | ***AtomicAGG*** | ***AGGREGATOR***

Due to the massive size of the entire historical BGP dataset, sub-sampling was incorporated by collecting five-day windows of data surrounding known historical BGP event timestamps. The known event timestamps served as domain knowledge

indicator to determine if unsupervised anomaly detection was accurately detecting events. Initial EDA ruled out supervised training for this use case due to the requirement of inconsistent methods to apply dependent variable across multiple events. However, supervised training could be applied after anomaly detection to determine root cause (malicious, misconfiguration, outage, etc.), but that will not be addressed in this project. Below is a sample of the events used as true positive metric and their raw data size:

- Nimda DoS Attack, 18 September 2001: 3,402,055 messages.
- Slammer Worm, 25 January 2003: 2,351,501 messages.
- TTNET BGP Misconfiguration, 24 December 2004: 434,671 messages.
- Mosco Blackout Google, 7 May 2005: 1,646,471 messages.
- TMnet BGP Misconfiguration, 12 June 2015: 19,982,317 messages.

4.2 Data Cleaning and Imputation

It was determined during EDA, that the parser output for the following fields were unusable null values of all observations in raw data. These columns were removed:

| Next_Hop | Local_Pref | MED | Community | AtomicAGG | AGGREGATOR

This resulted in a filtered raw data set approximately 28 million processed records.

	PROTOCOL	RECORD	TIMESTAMP	TYPE	PEERIP	PEERAS	PREFIX	AS_PATH	ORIGIN
2529	BGP4MP	901	01/26/03 23:59:54	A	192.65.185.144	6893	207.180.0.0/18	6893 3561 209 1784 1784 17014 7023	IGP
2530	BGP4MP	902	01/26/03 23:59:54	A	192.65.185.144	6893	63.136.98.0/24	6893 3561 1239 20240	IGP
2531	BGP4MP	903	01/26/03 23:59:54	A	192.65.185.144	6893	217.198.64.0/20	6893 12541 1273 702 16272 16272 13270 20769	IGP
2532	BGP4MP	904	01/26/03 23:59:54	A	192.65.185.144	6893	217.27.192.0/20	6893 12541 1273 702 16272 16272 13270	IGP
2533	BGP4MP	904	01/26/03 23:59:54	A	192.65.185.144	6893	217.28.224.0/20	6893 12541 1273 702 16272 16272 13270	IGP

Fig. 5. Filtered Raw Data Sample

Table 3. Raw Data Dictionary

Column Name	Description	Values/data type
PROTOCOL	Protocol of the message	BGP4MP
RECORD	Unique record ID	int
TIMESTAMP	timestamp	datetime
TYPE	UPDATE message type	A: Announcement W: Withdraw STATE: No routing change
PEERIP	Peer IP address	string
PEERAS	Peer AS number	string
PREFIX	IP address prefix	string
AS_PATH	sequence of AS path	string of AS numbers separated

	segments	by space
ORIGIN	Origin of the path information	IGP: NLRI is interior to the originating AS EGP: NLRI learned via the EGP protocol INCOMPLETE: NLRI learned by some other means

Next, the AS_PATH column was used to extract a new column showing the length of the AS path for that message. The remaining fields were then grouped by timestamp in 15-minute bins to create feature statistics based on computations during the time interval selected. NA values for these grouped bins were imputed to zero as the NA value just indicted none were counted in that timespan.

Categorical features were expanded to include numeric counts by each class per time window. Statistical features were derived from the AS length for each time window. The resulting data frame for each 5-day event subsample is now pared down to 480 rows with 11 features as shown below with feature descriptions in the following section.

```
dfPivot4.tail(5)
```

	TIMESTAMP	PROTOCOL-BGP4MP	TYPE-A	TYPE-STATE	TYPE-W	ORIGIN-EGP	ORIGIN-IGP	ORIGIN-INCOMPLETE	AS_L_mean	AS_L_max	AS_L_std
475	2015-06-14 22:45:00	12519	11705	42	772	0.0	11067.0	638.0	5.078521	27	2.609846
476	2015-06-14 23:00:00	19461	18085	48	1328	0.0	17209.0	876.0	4.624942	20	2.312567
477	2015-06-14 23:15:00	15788	14442	42	1304	0.0	13661.0	781.0	5.016152	30	2.881794
478	2015-06-14 23:30:00	14271	13108	48	1115	0.0	12556.0	552.0	5.448882	21	2.928684
479	2015-06-14 23:45:00	18505	17540	42	923	1.0	16657.0	882.0	5.138179	22	2.405697

Fig. 6. Cleaned Data Output Columns Example

In addition to the grouped features above, max prefix duplicate counts and grouping the messages by peer IP was added. This separate feature subset contains over 7000 rows with 13 features as shown below.

	TIMESTAMP	AS_LENGTH-max	AS_LENGTH-mean	AS_LENGTH-std	ORIGIN-EGP	ORIGIN-IGP	ORIGIN-INCOMPLETE	PEERIP	PREFIX_max_value_count	PROTOCOL-BGP4MP	TYPE-A	TYPE-STATE	TYPE-W
7030	2015-06-10 14:30:00	1.0	1.000000	0.00000	0.0	1.0	0.0	192.65.185.130	1.0	1.0	1.0	0.0	0.0
7031	2015-06-11 13:15:00	1.0	1.000000	0.00000	0.0	19.0	0.0	192.65.185.130	2.0	19.0	19.0	0.0	0.0
7032	2015-06-11 08:00:00	1.0	1.000000	0.00000	0.0	0.0	0.0	193.0.19.22	2.0	8.0	0.0	8.0	0.0
7033	2015-06-11 13:30:00	1.0	1.000000	0.00000	0.0	0.0	0.0	193.0.19.22	2.0	8.0	0.0	8.0	0.0
7034	2015-06-12 14:45:00	1.0	0.666667	0.57735	0.0	2.0	0.0	2001:7f8:1c:24a::22f:1	3.0	3.0	2.0	0.0	1.0

Fig. 7. Grouped by IP Data Output Columns Example

Both the ‘group by IP’ and ‘group by all’ datasets were used for modeling to not only compare models, but also to compare different approaches at feature engineering.

4.3 Feature Engineering

The volume features are generated across each time interval as bulleted below with some of the bullets highlighting multiple features based on different statistical measures:

- Count of announcements, withdrawals, and state message types over time.

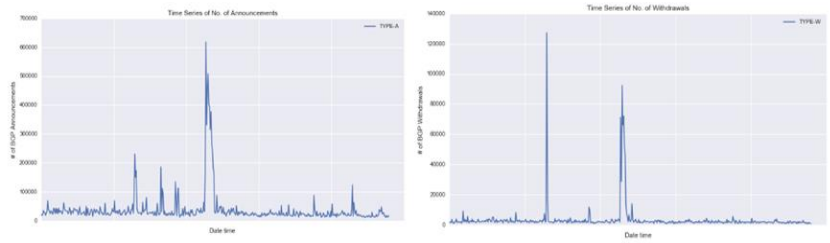


Fig. 8. Count of Announcements (left) and Withdrawals (right) vs. Time

- Count of messages originated from Interior Gateway Protocol (IGP), Exterior Gateway Protocol (EGP), and incomplete sources over time.

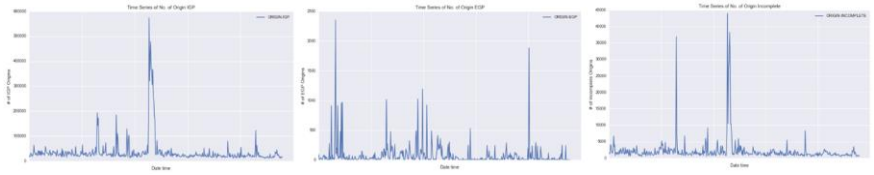


Fig. 9. Count of EGP (left), IGP (middle), and incomplete (right) vs. Time

- Average, maximum, and standard deviation of AS PATH steps (number of space delimited ASes in the AS PATH field) over time.

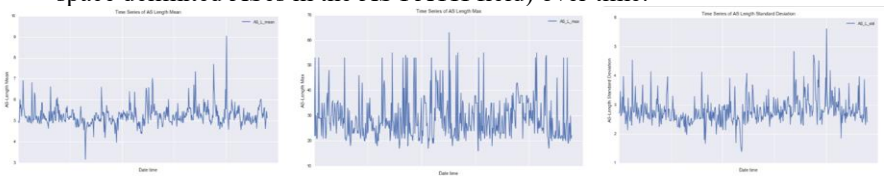


Fig. 10. AS Length Mean (left), Max (middle), SD(right) vs. Time

- Maximum count of announcements and withdrawal of unique NLRI prefixes during prescribed time interval.

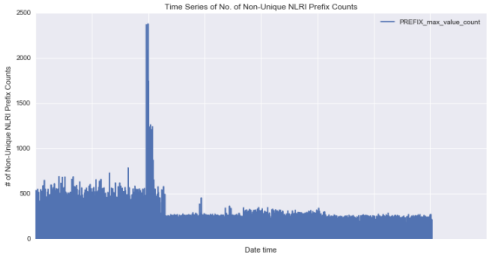


Fig. 11. maximum NLRI Prefix value counts vs. Time

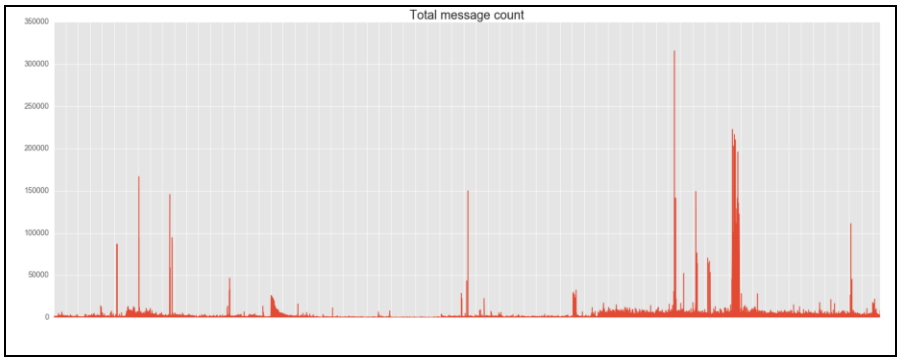
It is worth mentioning that some of these counts appear to correlate with event occurrences and anomaly detection may be possible with just a high pass filter. To get a fuller picture, however, will continue with cluster based unsupervised anomaly detection [14].

4.4 Data Normalization

Data normalization is an important part of data preparation when we are dealing with data features that have different ranges or units.

Considering the factors of internet traffic growth and span of sampled dates across 15 years, we need data normalization when comparing across these spans. Features related to message counts need to be scaled among subsets to reduce the bias caused by uneven data. Min-max scaling is applied for the normalization. (Figure 12: Total massage count: before and after min-max scaling)

After min-max scaling, subsets are merged together. We use standardization on the data to scale all features. A complete dataset is ready for analysis.



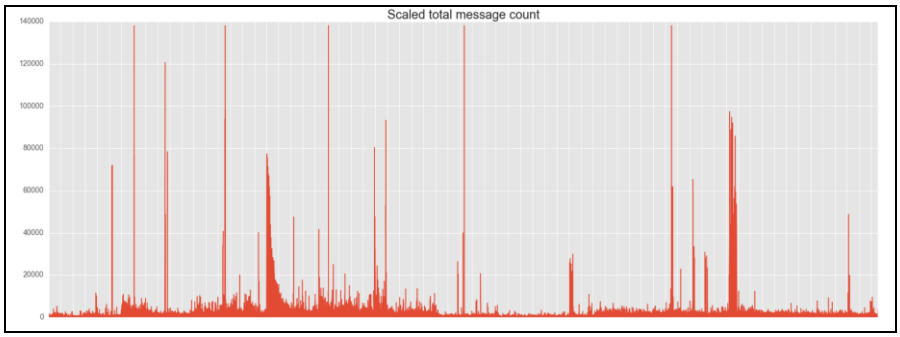


Fig. 12. Total massage count: before and after min-max scaling

5 Anomaly Detection

Clustering based anomaly detection was utilized with known event timestamps to compare models, tune parameters, and decide feature engineering. The clustering models were unsupervised, but the known event timestamps help act as a domain knowledge agent to determine if the anomaly detection was accurate or not for each pass. This was attempted using K-means and DBSCAN clustering.

Silhouette Coefficient

Silhouette analysis is used in the model evaluation.

The silhouette coefficient is a combination of two distances:

a. The mean distance between point i to other points within the same cluster where i is assigned.

b. The mean distance between point i to other points of the nearest cluster that i is not a part of.

$$\text{Silhouette Coefficient} = \frac{(b-a)}{\max(a,b)}$$

Silhouette coefficient shows how the points are close to the other points in the same cluster and how the points are distance from other clusters.

The range of silhouette coefficient is from -1 to 1, 1 is the best result and -1 is the worst. Score of 0 indicates overlapping clusters.

5.1 Unsupervised *k*-Means Clustering

The *k*-means clustering algorithm is widely used in many fields including anomaly detection. It creates 'k' similar clusters of data points. Data instances that fall outside of these groups could potentially be marked as anomalies. Before we start *k*-Means clustering, we used elbow method to determine the optimal number of clusters.

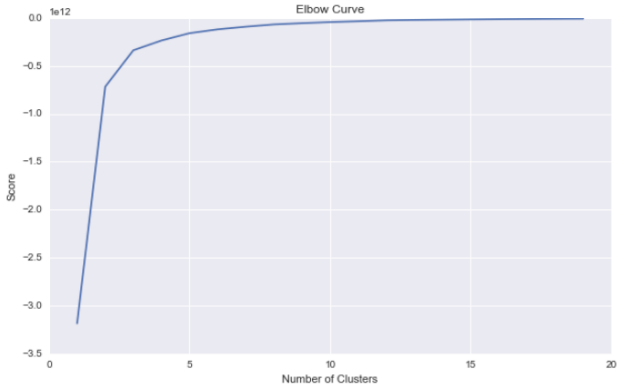


Fig. 13. K-means Cluster Elbow Curve

Although the elbow curve shows marginal improvement after 9, we selected 14 clusters for this project based on trial and error related to false positive counts. The underlying assumption in the clustering-based anomaly detection is that if we cluster the data, normal data will belong to clusters while anomalies will not belong to any clusters or belong to small clusters. We use the following steps to find and visualize anomalies [15].

- Calculate the distance between each point and its nearest centroid. The biggest distances are considered as anomaly.
- Set up a scoring method to provide information to the algorithm about the proportion of the outliers present in our data set.
- Calculate 'number of outliers' parameter using scoring method.
- Set threshold as the minimum distance of these outliers.
- The anomaly result will be a data field that contains the above method Cluster (0:normal, 1:anomaly).

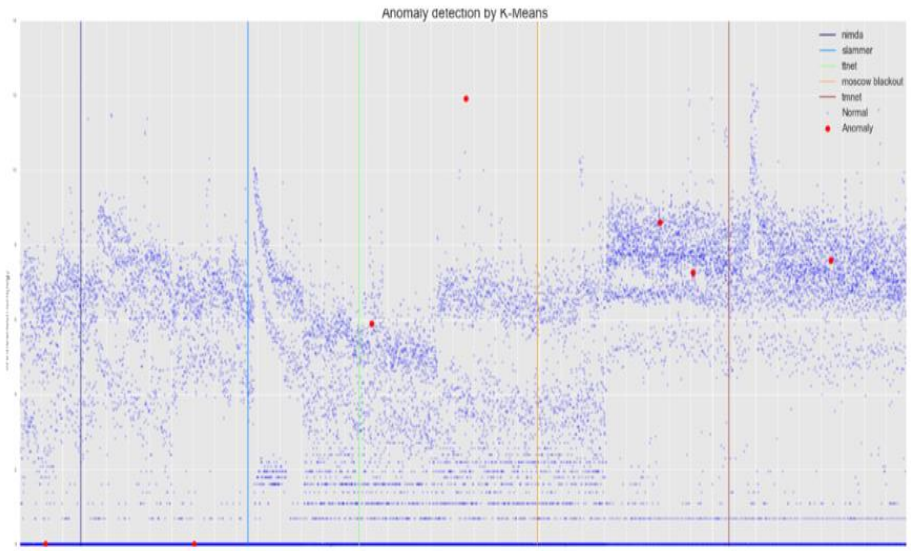


Fig. 14. Anomaly Detection Using *k*-Means Clustering

The results are a detection of 7 anomalies across 5 known event periods. The anomalies are shown on the graph as the red dots and the graph encompasses 25 days of normalized event activity. The silhouette score for this output was 0.652 and the number of clusters used was 14. This is a very good result as each anomaly was detected and the low number of actual anomalies listed shows an aversion to false positives. The model and settings used here are also resilient to over-selecting anomalies during 'quiet times', but still did predict a few false positives.

5.2 Unsupervised DBSCAN Clustering

DBSCAN, or density based spatial clustering of applications with noise, is a clustering algorithm that finds core samples of high density and expands clusters from them. It is good for data which contains clusters of similar density. Compared to centroid-based clustering like *k*-Means, density-based clustering works by identifying "dense" clusters of points, allowing it to learn clusters of arbitrary shape and identify outliers in the data.[16]

- Two main parameters of DBSCAN need to be set and then thresholding applied to their results to indicate anomalies:
    - eps - The maximum distance between two data points to be considered in the same neighborhood.
    - min_samples - The number of samples (or total weight) in a neighborhood for a point to be considered as a core point. [17]
    - Set threshold as the proportion of noise points.
    - Select eps and min_samples based on Silhouette scores and proportion of noise points
- Data points labeled as noise points by DBSCAN are considered anomaly.

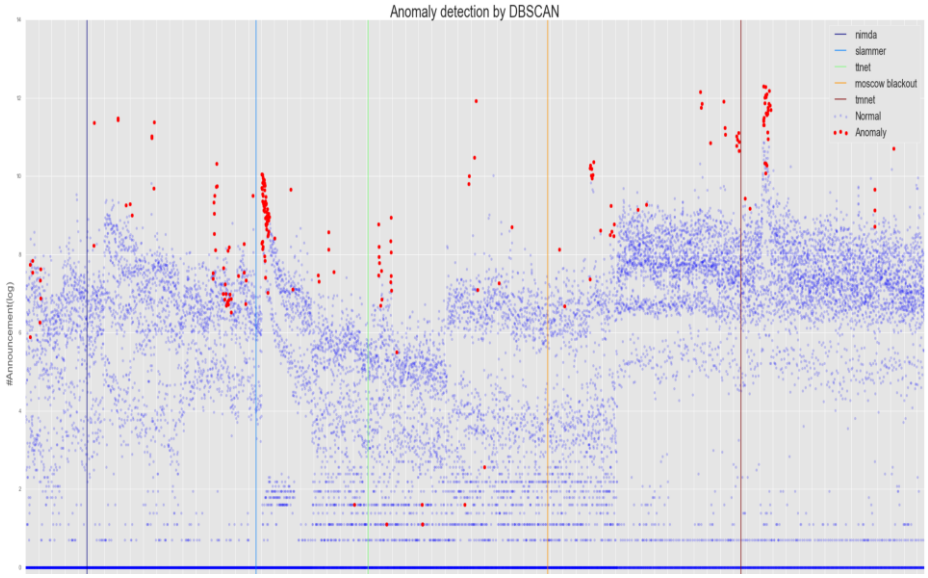


Fig. 15. Anomaly Detection Using DBSCAN Clustering

DBSCAN successfully detected anomalies during these event timelines. A total of 207 obeservations are detected as anomalies by DBSCAN with settings of eps = 3.5 and min_samples = 80. The Silhouette Coefficient is 0.836. DBSCAN is less resilient to outliers than *k*-means during non-event times. [18]

6 Machine Learning Pipeline Proposal

Integrating machine learning models into big data pipeline would ultimately be the goal of implementing a BGP anomaly trigger to make the insights actionable in near-real time. Up to this point in the document, the steps in the cluster analysis engine development on Figure 1 have been performed. The next steps to implement machine learning pipeline as shown on the bottom of Figure 1 would be as follows. The following steps are a proposal for future work in this domain. The assumption is a big data architecture is in place. The high-level steps are as follows:

1. Data is collected via job scheduler (cron, Oozie, etc.).
2. Script to create features (reproduce static feature creation output exactly) is implemented (Python, Spark, Flink).
3. Features pushed to storage for future review/use (HDFS, etc.).
4. Script to apply model(s) (Python, Spark, Flink).
    a. Also include revision control application to serialize the model with versioning.
    b. Chance to have feedback loop to parameterize model based on latest results (future works).

5. Publish anomalies (Kafka topic, HDFS, etc.).
6. Dashboard, Trigger (email, API to other system, etc.) each anomaly or groups of anomalies (ELK stack, Splunk, etc.).

7 Ethics

This project is not deeply affected by ethics issues as the BGP message data used is made available already for public use. If we had used private BGP syslog data, that information may have had specific customer IP addresses that would have required masking or concealing before statistical analysis. If using internal router logs, a private company could correlate neighbor IP information with external data to create many different BGP related triggers just based on correlations and high pass filters. These could include but are not limited to single customer all circuits idle, single router/card/interface all circuits idle, idle to established continuous flapping, etc. The risk of reverse engineering the IP addresses from the masks may carry too much of a negative to use this information outside of internal company systems.

8 Conclusions

The work covered in this paper has shown collection, parsing, feature engineering of public BGP message data and developing statistical/temporal features to apply unsupervised learning techniques. The most effective model for this iteration was k-means with a silhouette score of .652 using 14 clusters. As an unsupervised learning model, domain knowledge and inspection of the results determined this. False positives remained too high with DBSCAN methods attempted.

Next steps can include implementing proposed machine learning pipeline to apply the optimal model and parameters to near real-time data with an output to identify and alert network operations center of probable BGP anomalies. Future work can focus on several areas. Modeling improvements to reduce false positives can be implemented via new data sources to correlate, different techniques such as supervised learning, neural networks, hidden Markov model pattern comparisons, and new feature engineering. Router syslogs with BGP session establishment messages could be a huge boost in possible triggered alerts, even ones simply based on high pass filters as the state machine for this process has several possible event outcomes even on individual circuit basis.

References

1. Bahaa Al-Musawi, Philip Branch, Grenville Armitage, "Recurrence behaviour of BGP traffic", Telecommunication Networks and Applications Conference (ITNAC) 2017 27th International, pp. 1-7, 2017
2. Al-Musawi, B., Branch, P., Armitage, G.: BGP anomaly detection techniques: a survey. IEEE Commun. Surv. Tutor. 19(1), 377–396 (2017)

3. Jeff Dole , Routing TCP/IP, Volumn II, CCIE Professional Development, Second Edition, 2016.
4. Demystifying BGP Session Establishments. Dec 14, 2017, Retrieved from https://learningnetwork.cisco.com/blogs/vip-perspectives/2017/12/14/demystifying-bgp-session-establishment
5. BGP Peers, BGP Sessions and BGP Messages. Retrieved from https://ipcisco.com/lesson/bgp-peers-bgp-sessions-bgp-messages/
6. Stephen T. Kent, BBN Technologies, Securing the Border Gateway Protocol - The Internet Protocol Journal, Volume 6 Number 3
Retrieved from https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-25/securing-bgp-s-bgp.html
7. Rick Kuhn, Kotikalapudi Sriram, Doug Montgomery, "Border Gateway Protocol Security – Recommendations of the National Institute of Standards and Technology" National Institute of Standards and Technology Special Publication 800-54 Natl. Inst. Stand. Technol. Spec. Publ. 800-54, 61 pages (July 2007)
8. Jianning Mai, Lihua Yuan, Chen-Nee Chuah , " Detecting BGP Anomalies with Wavelet", Retrieved from https://www.ece.ucdavis.edu/~chuah/rubinet/paper/2008/noms08-BAlet.pdf
9. Dejing Dou, Jun Li, Han Qin , Shiwoong Kim, Sheng Zhong, "Understanding and Utilizing the Hierarchy of Abnormal BGP Events", Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.230.5203&rep=rep1&type=pdf
10. Route Views Project http://www.routeviews.org/routeviews/
11. Ripe Network Coordination Center: https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris
12. RFC 6396, Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format, IETF Standard https://tools.ietf.org/html/rfc6396
13. https://github.com/YoshiyukiYamauchi/mrtparse.git
14. Marijana Cosovic, Slobodan Obradovac, Ljiljana Trajkovic, "Using Databases for BGP Data Analysis" Unitech International Science Conference, November 21-22, 2014
15. J. Mai, L. Yuan and C. Chuah, "Detecting BGP anomalies with wavelet," NOMS 2008 - 2008 IEEE Network Operations and Management Symposium, Salvador, Bahia, 2008, pp. 465-472.
16. Junjie Wu, Advances in K-means Clustering: A Data Mining Thinking, July 10, 2012
17. Sayak Paul, DBSCAN: A Macroscopic Investigation in Python, August 3rd, 2018 . Retrieved from https://www.datacamp.com/community/tutorials/dbscan-macroscopic-investigation-python
18. scikit-learn Machine Learning in Python, API Reference, clustering, DBSCAN, Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html