International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Visualizing BGP Updates through Ladder Diagrams

Vishwas A Deshpande[1], Shwetha Baliga[2]

[1]*Student, Department of Electronics and Communications Engineering, RV College of Engineering, Bengaluru, India*

[2]*Assitant Professor, Department of Electronics and Communications Engineering, RV College of Engineering, Bengaluru, India*

Abstract - *Almost 90% of all traffic on the internet is routed using the Border Gateway Protocol, or BGP for short, which is a standard exterior gateway protocol (EGP) designed to exchange routing and reachability information among autonomous systems (AS) on the Internet. BGP is the protocol of choice for all major ISPs in the world. It is classified as a path vector protocol, meaning that neighboring nodes in a computer network exchange routing information to build routes. BGP is highly scalable; However, with increase in network size, troubleshooting issues arising in the implementation of BGP becomes laborious due to a multitude of reasons. Troubleshooting issues in BGP mainly involves pulling up system logs which usually span thousands of lines from the routers in the network, and manually searching for the relevant logs, identifying the issue, and then taking corrective action. Furthermore, even when BGP is functioning correctly, identifying the reason why a particular route was installed in a router's routing table is not intuitive, since one has to use the command line interface on the routers to get this information. The objective of this project work is to develop a tool which can help make troubleshooting and monitoring of BGP networks more intuitive and easy for the network engineers involved through the use of Ladder Diagrams or Sequence Charts, which show the flow of information between different stages of a process, and also filter system logs automatically to present only the relevant logs. The use of the tool developed through this project work greatly reduces troubleshooting speed, hence reducing the workload on network engineers, and greatly reducing network down time.*

Key Words: Border Gateway Protocol, Logs, Troubleshooting, Network Serviceability, Routing Protocol, Internet Service Provider

1. INTRODUCTION

Border Gateway Protocol (BGP) is an Internet Engineering Task Force (IETF) standard, and the most scalable of all routing protocols. BGP is the routing protocol of the global Internet, as well as for Service Provider private networks. BGP has expanded upon its original purpose of carrying Internet reachability information, and can now carry routes for Multicast, IPv6, VPNs, and a variety of other data.

When BGP is configured incorrectly, it can cause massive availability and security problems, as Google discovered in 2008 when its YouTube service became unreachable to large portions of the Internet. What happened was that, in an effort to ban YouTube in its home country, Pakistan Telecom used BGP to route YouTube's address block into a black hole. But, in what is believed to have been an accident, this routing information somehow got transmitted to Pakistan Telecom's Hong Kong ISP and from there got propagated to the rest of the world. The end result was that most of YouTube's traffic ended up in a black hole in Pakistan. More sinisterly, 2003 saw a number of BGP hijack attacks, where modified BGP route information allowed unknown attackers to redirect large blocks of traffic so that it travelled via routers in Belarus or Iceland before it was transmitted on to its intended destination.

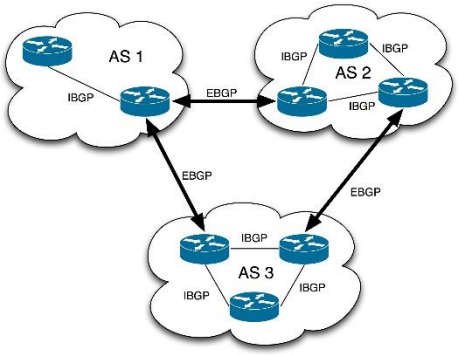

Fig-1: Example BGP Topology

When network engineers have to tackle any problems regarding either the configuration or maintenance of BGP on network routers, they use the logs generated by BGP debugging on the routers to gain detailed information about different events and updates with regard to BGP. However, a typical log file may exceed millions of lines, and it becomes very

difficult and time consuming to manually parse through the logs. Also, the logs generated are not intuitive to the network engineers to grasp.

This project work intends to create a method to visualize important BGP update information extracted from the BGP logs, in the form of ladder diagrams. Ladder diagrams are visual representations of the different stages/events of BGP in the form of a ladder, which make it very intuitive to the network engineer trying to troubleshoot a BGP issue.

1.1 BGP Router ID

BGP uses a router ID to identify BGP-speaking peers. The BGP router ID is a 32-bit value that is often represented by an IPv4 address. By default, the Cisco software sets the router ID to the IPv4 address of a loopback interface on the router. If no loopback interface is configured on the device, the software chooses the highest IPv4 address configured on a physical interface of the device to represent the BGP router ID. The BGP router ID must be unique to the BGP peers in a network.

1.2 BGP Neighborships

A BGP-speaking device does not discover another BGP-speaking device automatically. A network administrator usually manually configures the relationships between BGP- speaking devices. A peer device is a BGP-speaking device that has an active TCP connection to another BGP-speaking device. This relationship between BGP devices is often referred to as a neighbor, but because this can imply the idea that the BGP devices are directly connected with no other device in between, the term neighbor will be avoided whenever possible in this document. A BGP speaker is the local device, and a peer is any other BGP-speaking network device. When a TCP connection is established between peers, each BGP peer initially exchanges all its routes—the complete BGP routing table—with the other peer. After this initial exchange, only incremental updates are sent when there has been a topology change in the network, or when a routing policy has been implemented or modified. In the periods of inactivity between these updates, peers exchange special messages called keep lives.

A BGP autonomous system is a network that is controlled by a single technical administration entity. Peer devices are called external peers when they are in different autonomous systems and internal peers when they are in the same autonomous system. Usually, external peers are adjacent and share a subnet; internal peers may be anywhere in the same autonomous system.

1.3 BGP Peer Session Establishment

When a BGP routing process establishes a peering session with a peer, it goes through the following state changes:

1) Idle—The initial state that the BGP routing process enters when the routing process is enabled or when the device is reset. In this state, the device waits for a start event, such as a peering configuration with a remote peer. After the device receives a TCP connection request from a remote peer, the device initiates another start event to wait for a timer before starting a TCP connection to a remote peer. If the device is reset, the peer is reset and the BGP routing process returns to the idle state.

2) Connect—The BGP routing process detects that a peer is trying to establish a TCP session with the local BGP speaker. Active—In this state, the BGP routing process tries to establish a TCP session with a peer device using the Connect Retry timer. Start events are ignored while the BGP routing process is in the Active state. If the BGP routing process is reconfigured or if an error occurs, the BGP routing process will release system resources and return to an Idle state.

3) Open Sent—The TCP connection is established, and the BGP routing process sends an OPEN message to the remote peer, and transitions to the OpenSent state. The BGP routing process can receive other OPEN messages in this state. If the connection fails, the BGP routing process transitions to the Active state.

4) OpenReceive—The BGP routing process receives the OPEN message from the remote peer and waits for an initial keepalive message from the remote peer. When a keepalive message is received, the BGP routing process transitions to the Established state. If a notification message is received, the BGP routing process transitions to the idle state. If an error or configuration change occurs that affects the peering session, the BGP routing process sends a notification message with the Finite State Machine (FSM) error code and then transitions to the Idle state.

5) Established—the initial keepalive is received from the remote peer. Peering is now established with the remote neighbor and the BGP routing process starts exchanging update message with the remote peer. The hold timer restarts when an update or keepalive message is received. If the BGP process receives an error notification, it will transition to the Idle state.

2.0 MSCGEN

Mscgen is the program used to generate Ladder Diagrams in this project, which is described in detail below.

2.1 Overview of Mscgen

Mscgen is a small program that parses Message Sequence Chart descriptions and produces PNG, SVG, EPS or server side image maps (ismaps) as the output. Message Sequence Charts (MSCs) are a way of representing entities and interactions over some time period and are often used in combination with SDL. MSCs are popular in Telecoms to specify how protocols operate although MSCs need not be complicated to create or use. Mscgen aims to provide a simple text language that is clear to create, edit and understand, which can also be transformed into common image formats for display or printing.
This program and the language it parses have been in- spired by Graphviz Dot, which provides a really good way to document State Transition Diagrams, data structures and directed graphs. Unlike Graphviz, this program does no clever layout operations or spline routing as this is not needed for MSCs, and so was much simpler to implement. Doxygen (version 1.5.2 onwards) also allows MSCs to be embedded directly in the same way that dot diagrams can be added to documentation, making it easy to improve Doxygen generated documentation through the use of message sequence charts.
Mscgen is licenced under the GPLv2. This covers use of the program sources but places no restriction on the usage of the tool itself or the diagrams it produces. Mscgen is written in ANSI-C, and uses the GD graphics library for PNG output. Mscgen can be built under Linux, Cygwin, and as a native Win32 application through Cygwin with the wonderful -no- Cygwin compile option. Since version 0.17 it uses autocon- f/automake, so should be reasonably automatic in building provided that the dependent packages (gcc, flex, bison, libgd- devel) are available. Others have previously reported success building the sources SunOS 5.8, Solaris 10, FreeBSD as well as Mac OS X.

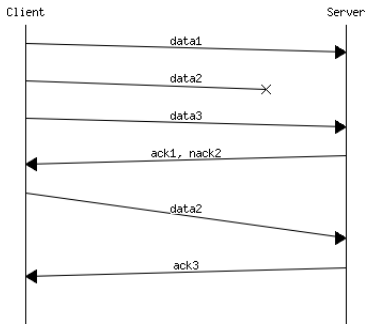

Fig -2: Ladder Diagram

Mscgen involves a very simple, human readable syntax to generate ladder diagrams. The code below is a typical example to generate such a ladder diagram through Mscgen.
As seen in the code above, each node is declared first, followed by each flow arrow declared separately using a very simple syntax that is human readable.
The output of the MSc code is seen in Fig. 3. It can be observed that Mscgen greatly cuts down the time and effort needed to generate such ladder diagrams by automating it.

```
# Fictional client-server protocol
msc {
 arcgradient = 8;

 a [label="Client"],b [label="Server"];

 a=>b [label="data1"];
 a-xb [label="data2"];
 a=>b [label="data3"];
 a<=b [label="ack1, nack2"];
 a=>b [label="data2", arcskip="1"];
 |||;
 a<=b [label="ack3"];
 |||;
}
```

3.0 METHODOLOGY

This project work aims to represent the BGP process visually using ladder diagrams; A brief overview of the methodology is presented here.

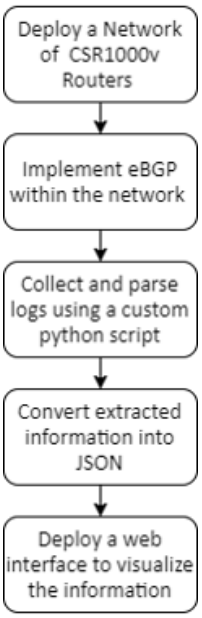

Fig -3: Methodology of development

As shown in Fig.3, the first step involves creating a network of virtual routers on an EXSi server for the purpose of implementing BGP and generating BGP advertisements and logs in the network, which forms the data on which our tool works.
In the next step, BGP is implemented in the network by
configuring the routers using the 'router bgp' command, which is an inbuilt command in Cisco IOS, Cisco's operating system for routers and switches. This enables BGP on the routers and with a few more lines of configuration generates all the data this project would need to work.
In the third step, custom python code written for this project
will establish a remote session with any one of the routers deployed, either over an SSH or Telnet link. Then, the tool will proceed to collect the logs generated by the router which would have been stored in a local buffer. Finally, the logs collected are parsed through, and relevant logs are stored in a JSON object, which helps the tool generate the required ladder Diagrams.

Finally, the last step involves the creation and deployment of a simple, clean UI which helps the users of this project to provide details needed by this tool, and to display the Ladder Diagram generated along with the relevant log files.
Each step is detailed in the following subsections.

3.1 Deploying the Network

The test network used to implement the first step involves creating a network of virtual routers on an EXSi server for the purpose of implementing BGP and generating BGP advertisements and logs in the network, which forms the data on which our tool works. The CSR1000v series of routers from Cisco was selected, owing to ease of deployment and the wide range of functionalities it supports. The CSR 1000V Series serves as a secure single-tenant router in a multitenant, shared-resource public cloud environment. It provides end-to- end managed connectivity. The network was deployed on an EXSi server, allowing easy control and remote access. The entire network is reachable via managements IPs which are a part of the Cisco subnet.

3.2 Implementing BGP in the Network

Once the network is deployed, BGP is configured in each of the networks to generate network advertisements and system log messages for BGP, which can be of various categories.
A sample configuration of BGP on one of the routers is shown below:

```
router bgp 45000
 router-id 172.17.1.99
 bgp log-neighbor-changes
 neighbor 192.168.1.2 remote-as 40000
 neighbor 192.168.3.2 remote-as 50000
 address-family ipv4 unicast
  neighbor 192.168.1.2 activate
  network 172.17.1.0 mask 255.255.255.0
  exit-address-family
 address-family ipv4 multicast
  neighbor 192.168.3.2 activate
  neighbor 192.168.3.2 advertisement-interval 25
  network 172.16.1.0 mask 255.255.255.0
  exit-address-family
 address-family ipv4 vrf vpn1
  neighbor 192.168.3.2 activate
  network 172.21.1.0 mask 255.255.255.0
  exit-address-family
```

The above configuration specifies BGP neighborship that need to be established with directly connected peers. The 'router bgp 45000' commands specifies the AS number this particular router will belong to. The

'neighbor' commands specifies the IP address of the directly connected interface of a BGP peer.

3.3 Collecting and Parsing Logs

In this step, custom python code written for this project will establish a remote session with any one of the routers deployed, either over an SSH or Telnet link. The Paramiko library is used to establish an SSH connection to the routers for remote access. Paramiko is a Python (2.7, 3.4+) implementation of the SSHv2 protocol, providing both client and server functionality. While it leverages a Python C extension for low level cryptography (Cryptography), Paramiko itself is a pure Python interface around SSH networking concepts. Once the remote connection is established, the tool will proceed to collect the logs generated by the router which would have been stored in a local buffer. For this, the tool sends the 'show logging' command to the router, which results in the router sending the contents of the logging buffer on the router to the tool.

```
*Feb 27 05:57:23.157: %BGP-3-NOTIFICATION_MANY: sent to 2 sessions 6/4 (Administ
*Feb 27 05:57:23.171: %BGP-5-ADJCHANGE: neighbor 10.104.123.2 Down User reset
*Feb 27 05:57:23.171: %BGP_SESSION-5-ADJCHANGE: neighbor 10.104.123.2 IPv4 Unica
*Feb 27 05:57:23.171: %BGP-5-ADJCHANGE: neighbor 10.104.123.3 Down User reset
*Feb 27 05:57:23.172: %BGP_SESSION-5-ADJCHANGE: neighbor 10.104.123.3 IPv4 Unica
*Feb 27 05:57:30.502: %BGP-5-ADJCHANGE: neighbor 10.104.123.3 Up
*Feb 27 05:57:34.410: %BGP-5-ADJCHANGE: neighbor 10.104.123.2 Up
*Feb 27 06:07:34.490: %SYS-5-CONFIG_I: Configured from console by cisco on vty0
*Feb 27 06:19:19.768: FIBtable: [v4:Default] Start of table refresh
*Feb 27 06:19:19.770: CEF: Delete 0 prefix 224.0.0.0/4
*Feb 27 06:19:19.771: FIBtable: [v4:Default] Repopulate from RIB start (epoch 3)
*Feb 27 06:19:19.771: FIBtable: [v4:Default] Repopulation succeeded (10 ndbs)
*Feb 27 06:19:19.772: FIBtable: [v4:Default] End of table refresh
*Feb 27 06:19:19.772: FIBtable: [v4:Default] Starting purge of table to epoch 3
*Feb 27 06:19:19.772: FIBtable: [v4:Default] Purged 0 prefixes from table (rc=No
*Feb 27 06:24:03.289: %OSPF-5-ADJCHG: Process 10, Nbr 10.20.3.3 on GigabitEthern
*Feb 27 06:24:05.312: %SYS-5-CONFIG_I: Configured from console by cisco on vty0
*Feb 27 07:01:21.061: %BGP-3-NOTIFICATION_MANY: sent to 2 sessions 6/4 (Administ
*Feb 27 07:01:21.076: %BGP-5-ADJCHANGE: neighbor 10.104.123.2 Down User reset
*Feb 27 07:01:21.076: %BGP_SESSION-5-ADJCHANGE: neighbor 10.104.123.2 IPv4 Unica
*Feb 27 07:01:21.077: %BGP-5-ADJCHANGE: neighbor 10.104.123.3 Down User reset
*Feb 27 07:01:21.077: %BGP_SESSION-5-ADJCHANGE: neighbor 10.104.123.3 IPv4 Unica
*Feb 27 07:01:27.980: %BGP-5-ADJCHANGE: neighbor 10.104.123.2 Up
*Feb 27 07:01:29.273: %BGP-5-ADJCHANGE: neighbor 10.104.123.3 Up
```

Fig -4: Typical logs

As shown in Fig.4, the logs generated by routers aren't very intuitive, and consume a lot of time to go through when a network engineer is trying to troubleshoot issues.

Once the logs are received, the custom python code written helps the tool parse through the logs, and collect all the logs which match:

- BGP Events
- BGP Updates
- Routing Table events
- CEF Table events

Then the tool proceeds to create JSON objects which are needed by Mscgen.

3.4 Creating JSON objects

Once the logs are filtered out, they need to be reshaped into a JSON object, with IP addresses of different destinations(networks) as the keys, with the logs for those networks as values. Furthermore, each destination is advertised by multiple neighbors, and hence the JSON object is in fact a multi-level dictionary. As shown in Fig.6, the JSON object is multi-level, and contains logs categorized by:

- Destination Prefix
- Neighbor ID
- Category: BGP, RT or CEF


```
{
  "10.20.1.0/24": {
    "bgp": [
      "",
      "Apr  2 20:16:03: BGP(0): local route 10.20.1.0/24 added gw 0.0.0
       .0\n",
      "Apr  2 20:16:03: BGP: topo global:IPv4 Unicast:base
       Remove_fwdroute for 10.20.1.0/24\n",
      "Apr  2 20:16:14: BGP(0): (base) 10.104.12.2 send UPDATE (format)
       10.20.1.0/24, next 10.104.12.1, metric 0, path Local\n",
      "Apr  2 20:16:14: BGP(0): 10.104.12.2 rcv UPDATE about 10.20.1.0
       /24 -- DENIED due to: AS-PATH contains our own AS; NEXTHOP is
       our own address;\n",
      "Apr  2 20:16:14: BGP(0): 10.104.14.4 rcv UPDATE about 10.20.1.0
       /24 -- DENIED due to: AS-PATH contains our own AS; NEXTHOP is
       our own address;\n",
      "Apr  2 20:16:45: BGP(0): 10.104.14.4 rcv UPDATE about 10.20.1.0
       /24 -- DENIED due to: AS-PATH contains our own AS;\n",
      "Apr  2 20:16:45: BGP(0): 10.104.12.2 rcv UPDATE about 10.20.1.0
```


Fig. 5: JSON Object Created

3.5 Web Interface

The tool consists of a web interface built using HTML, CSS and Cisco UI Kit. The Cisco UI kit and pattern library is a style guide governing HTML markup at the presentation layer. It contains no JavaScript components. Instead, it focuses on HTML and CSS and leaves the choice of JavaScript toolkit up to the individual development teams. This library is a collaboration between Cisco Brand and Cisco Engineering. It is modern, lightweight, responsive and programmable presentation layer that can be used to style HTML primitive elements and UI-framework independent. This project work mainly involves the use of the Cisco UI Kit to develop the web interface. The UI consists of a simple form which asks for the Router to connect to over SSH, and the credentials as well. Once the connection is established, the prefixes received by that particular router are displayed.

Fig.-6: Prefix List Generated

As shown in Fig.6, the received prefixes are displayed, clicking one of which generates the Ladder Diagram for that prefix, followed by displaying the relevant logs. The ladder diagram generated and the filtered log display are discussed in the next section.

4.0 RESULTS

The tool consists of a simple webpage which asks for the IP address of the device to SSH into, upon submitting which, the backend python script runs, connecting over SSH to the device, collecting BGP details, and extracts the BGP prefixes advertised to that device, which are then displayed to the user. The whole project runs as a Flask app and has been Dockerized to run on any platform. The base OS is Ubuntu: 18.04, selected for its vast support and stability. The Docker file is very simple, with just a few instructions to install the dependencies required namely Flask, Flask Forms and Paramiko.

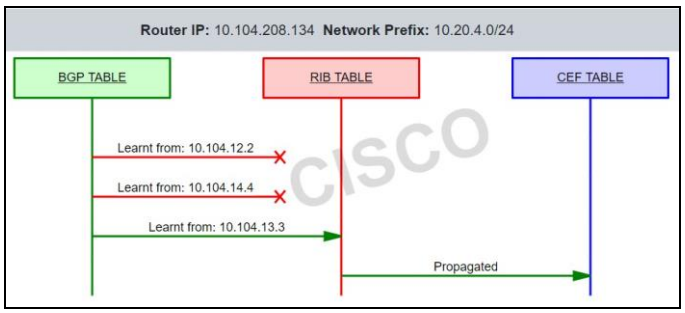

Fig-7: Ladder Diagram Generated By the Tool

As shown in Fig.6, the prefixes received are displayed for further input. Upon clicking on a network prefix, the tool then connects to the device again, collects logs, parses them for relevant ones, and then also generates a Ladder Diagram from all the information collected, as shown in Fig.8.

The Ladder Diagram consists of three nodes, as can be seen:

1) **BGP TABLE:** It contains the network layer reachability information (NLRI) learned in compliance with BGP and NLRI attributes (path attribute, PA) corresponding to these path. Essentially, NLRI is a prefix and its length. BGP table contains all the routes from all the neighbors, several routes to the same network with different attributes.

2) **RIB TABLE:** It is a data table stored in a router or a network host that lists the routes to particular network destinations, and in some cases, metrics (distances) associated with those routes. The routing table contains information about the topology of the network immediately around it.

3) **CEF TABLE:** It is conceptually similar to a routing table or information base. It maintains a mirror image of the forwarding information contained in the IP routing table. When routing or topology changes occur in the network, the IP routing table is updated, and these changes are reflected in the CEF Table. It also maintains next-hop address information based on the information in the IP routing table.

The Ladder Diagram shows the flow of network prefixes from the BGP table to the RIB table and then to the CEF table; It is to be noted that only the best path is propagated, which is decided by a number of BGP attributes like local preference and weight, which are configured in the router.

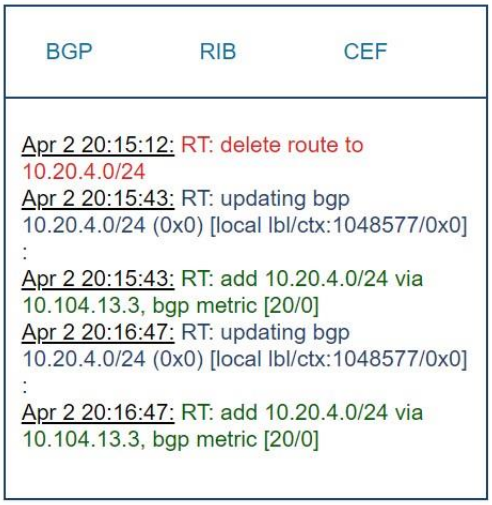

Fig-9: Filtered Logs

The tool also filters out relevant logs and displays them in a table on the UI, as can be seen in Fig.9. It is also color coded. This was done using some regex matching in python followed by styling in the HTML code. Also, clicking on any of the propagation arrows in the Ladder Diagram filters the logs further to only display the ones from that particular BGP neighbor.
The tool built in this project work was successful in achieving all the objectives:

- Make the troubleshooting process more intuitive.
- Cut down troubleshooting time.
- Decrease network downtime.
- Building a new learning mechanism as well.

6.0 CONCLUSION

BGP is the backbone protocol of the internet. Deploying BGP in your network ensures the security of the routing information exchange and the stability of the network. troubleshooting the flow of BGP updates inside a router is very difficult and time consuming as the log messages are innumerable.
This project was to intuitively help in troubleshooting by creating a tool to visualize the flow of BGP updates (including the best path selection) inside a router (prefix propagation from BGP all the way up to CEF) using ladder diagrams and display only the relevant log messages.
This project was done with the motive to enhance the support serviceability effort, which is an important arm of any net- working enterprise.
The use of the tool developed through this project work greatly reduces troubleshooting speed, hence reducing the work load on network engineers, and greatly reducing network down time.

7.0 FUTURE SCOPE

This project is a step in the direction of automation of network maintenance and troubleshooting, which is the latest trend in networking. Hence there exist many extensions to this project which are possible, either expanding its scope, or adding more functionality in the current scope itself. Some of the ideas which can be implemented are:

- Adding further functionality for radioactive tracing of BGP updates.
- Incorporating features to also enable users to use this as a learning tool.
- Expanding support to various other exterior gateway protocols
- Incorporating anomaly detection in BGP networks.

REFERENCES

[1] Y. Rekhter, T. Li, and S. Hares,"A Border Gateway Protocol 4 (BGP-4)", RFC 4271, Jan. 2016.

[2] M. Caesar and J. Rexford, "BGP routing policies in ISP network", *IEEE Network*, vol. 19, no. 6, pp. 5–11, Nov.–Dec. 2005.

[3] Seitan, A. (2015). *Nixtrain-CCNA Lab Guide Nixtrain* 1st Edition Full Version. Bandung: CV. Nixtrain Infomatica. 271–350.

[4] B. Quoitin and O. Bonaventure, "A Survey of the Utilization of the BGP Community Attribute," expired *Internet Draft draft-quoiting-bgp-comm-* survey00.txt, Feb. 2002.

[5] Y. Yang et al., "On Route Selection for Interdomain Traffic Engineering," *IEEE Network*, Jan. 2011

[6] N. Feamster, J. Winick, and J. Rexford, "A Model of BGP Routing for Network Engineering," *Proc. ACM SIGMETRICS*, June 2004.

[7] J. Bartlett, "Optimizing Multi-homed Connections," *Bus. Commun. Rev.*, vol. 32, no. 1, Jan. 2002, pp. 22–27.

[8] O. Nordstrom and C. Dovrolis, "Beware of BGP Attacks," *ACM SIGCOMM Comp. Commun. Rev.,* Apr. 2004.

[9] N. Feamster and H. Balakrishnan, "Detecting BGP Configuration Faults with Static Analysis," *Proc. Networked Sys. Design and Implementation*, May 2005.

[10] T. Griffin, A. Jaggard, and V. Ramachandran, "Design Principles of Policy Languages for Path Vector Protocols," *Proc. ACM SIGCOMM*, Aug. 2013.

[11] S. Kent, C. Lynn and K. Seo, Secure Border Gateway Protocol (SBGP), *IEEE Journal on Selected Areas in Communications*, Vol. 18, No 4, pp 582-592, April 2000.

[12] Y. Wang, M. Schapira, and J. Rexford, "Neighbor-specific BGP: more flexible routing policies while improving global stability," *2009 SIG- METRICS.*

[13] D. Walton et al., "Advertisement of multiple paths in BGP," *IETF draft*, Dec. 2012.

[14] J. Uttaro et al., "Best practices for advertisement of multiple paths in BGP," *IETF draft,* Jul. 2010