1-1-2012

Policy Disputes In BGP: Analysis, Detection And Proposed Solution

Baha U. Kazi
Ryerson University

POLICY DISPUTES IN BGP: ANALYSIS, DETECTION AND PROPOSED SOLUTION

by

Baha Uddin Kazi

M.Sc in Computer Science

National University Bangladesh, 2003

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Masters of Applied Science

in the Program of

Computer Networks

Toronto, Ontario, Canada, 2012

Author's Declaration

POLICY DISPUTES IN BGP: ANALYSIS, DETECTION AND PROPOSED SOLUTION

Master of Applied Science, 2012, Baha Uddin Kazi, Computer Networks, Ryerson University.

ABSTRACT

Border Gateway Protocol (BGP) is the de-facto inter-domain routing protocol in the Internet, which involves exchange of routing information among the ASes that is used by the routers in an AS to compute paths to destination address blocks or prefixes in the Internet. The BGP is a path-vector and policy-based routing that allows each AS to independently define a set of local policies for route selection. However, since routes are selected based on local policies of the ASes, it might cause global conflicts or network topology disputes among the ASes. In this thesis, first we present a survey and in-depth analysis of the existing available solutions and their implementations for the BGP policy induced faults. Then, we discuss the tool we have developed for identifying the BGP faults or route instability within an autonomous system due to local policy. Finally, we propose a new solution to detect and eliminate the route oscillation in the BGP best path selection process due to local policy conflicts among ASes. We also present the test cases that we developed using BGP simulator for simulating oscillation faults in the test-bed and then discuss their results.

Acknowledgement

First of all, I would like to thanks Almighty Allah who gave me the ability to accomplish this thesis. I would like to address my very special thanks to my supervisor, Dr. Muhammad Jaseemuddin for his guidance and assistance throughout this research. I am forever grateful to him for all that he has done for me to successfully complete this research. I would also like to thank Dr. Bobby Ma for his advice and tremendous support. I am very much thankful to my research team in the lab: Khalid A. Hafeez, Frank, Xiaoli Lisa, Jeremy Udit, Rashid faruqui, Zainab shamsi and Arseny Taranenko for their kind help and advice. I also acknowledge the support of Solana Networks and FedDev ARC project to provide financial and material support in accomplishing my research. Finally I want to pay my gratitude to my family for their support and encouragement.

Table of Contents

ABSTRACT iii

Acknowledgement iv

Table of Contents v

List of Abbreviation xi

Chapter One 1

Introduction 1

1.1 Introduction 1

1.2 Thesis Overview 3

Chapter Two 4

Background and Literature Review 4

2.1 Internet Architecture 4

2.2 BGP Overview 6

2.3 BGP Attributes 9

2.4 Routing Policy and Decision Process 12

2.5 Related Works 15

Chapter Three 16

Policy Induced Topology Disputes and Proposed Solutions 16

3.1 BGP Policy Disputes 17

3.1.1 BGP Policy Induced Oscillation 17

3.1.2 Dispute Wheel 21

3.1.3 Policy-Induced Route Non-Determinism 22

3.1.4 MED oscillation problem 24

3.2 Prior Solutions for Policy Induced Route Instability 26

3.2.1 Stable Path Problem (SPP) 26

3.2.2 Simple Path Vector Protocol (SPVP) 28

3.2.3 Example 29

3.2.4 Solution Using Precedence Metric 30

3.2.5 Enforcing Convergence 33

3.2.6 Proposed Solution for Inter-Domain Policy Disputes 35

3.3: Implementation of Proposed Solution for Inter-Domain Policy Disputes 39

3.3.1 Test Case One 39

3.3.2 Test Case Two 42

3.3.3 Test Case Three 44

3.4 Comparison between original simulator and New simulator 46

Chapter Four 47

Instability Detection Tool, Test Cases and Results 47

4.1 Routing Data Analysis from public source 48

4.2 Test Network Setup 49

4.3 BGP Simulator 50

4.4 Proposed Fault Detection Algorithm 51

4.5 Architecture of the Software 53

4.6 Software Development 54

4.7: Test Cases to check The Policy Induced Topology Disputes 55

4.7.1: Test Case One: 55

4.7.2: Test Case Two 62

4.7.3: Test Case Three With Different Next Hop 71

4.7.4: Test Case Four for Non-determinism 78

4.8: Network Setup and Fault Test For Real World Data 83

Chapter Five 91

Conclusion and Future Work 91

Appendix A 93

BGP Step-by-Step Fault Test Methodology 93

Step by Step BGP Fault Detection Using Tool 94

References 101

Table of Figures

Figure 2.1: General Routing Architecture 6

Figure 2.2: BGP Route Selection Method 14

Figure 3.1: A triangular network and Local policy 18

Figure 3.2: Node by Node Observation of Oscillating Gadget 19

Figure 3.3: Observation of Oscillating Gadget whole topology 20

Figure 3.4: Dispute Wheel 21

Figure 3.5: Network with business relation and backup link 22

Figure 3.6: Network with intended path 23

Figure 3.7: Network with unintended path 23

Figure 3.8: Oscillation due to MED 25

Figure 3.9: Stable Path Problem with solution - Good Gadget 29

Figure 3.10: Stable Path Problem with no solution - Bad Gadget 29

Figure 3.11: Sequence of path assignment for Bad Gadget 30

Figure 3.12: A triangular network and Local policy 31

Figure 3.13: Step by Step solution of the problem in fig. 3.11 according to the Precedence Metric 32

Figure 3.14: Step by Step solution of the problem in fig. 3.11 according to the Cobb & Musunuri 34

Figure3.15: Our Proposed Algorithm 36

Figure 3.16: Step by Step solution of the problem in figure 3.11 according to our proposed solution 37

Figure 3.17: Solution of the Oscillation If we start from V_1 38

Figure 3.18: Test Case One Topology and Configuration 41

Figure 3.19: Output of Test Case One 41

Figure 3.20: Test Case Two Topology and Configuration 43

Figure 3.21: Output of Test Case Two..43

Figure 3.22: Test Case Three Topology and Configuration...45

Figure 3.23: Output of Test Case Three 46

Figure 3.24: Comparison between our algorithm and original simBGP 46

Figure 4.1: Overview of The Research Study 47

Figure 4.2: Test Network...49

Figure 4.3: Topology Disputes due to Local Policy Using BGP Simulator 50

Figure 4.4: Route Instability Detection Algorithm 52

Figure 4.5: Architecture of the route instability detection tool within an AS 53

Figure 4.6: Program Structure of route instability detection algorithm...................................54

Figure 4.7: Topology for Test Case One manipulating weight attribute...................................56

Figure 4.8: BGP RIB for Test Case One 57

Figure 4.9: BGP Sample Update for Test Case One 60

Figure 4.10: Database Table converted from XML BGP Data 60

Figure 4.11: Program output of Test Case One 61

Figure 4.12: Test Case Two 63

Figure 4.13: BGP RIB of Test Case Two...66

Figure 4.14: BGP Sample Updates of Test Case Two 69

Figure 4.15: Database Table for Test Case Two..70

Figure 4.16: Program Output of Test Case Two 70

Figure 4.17: Topology for Test Case Three manipulating Local Preference attribute 72

Figure 4.18: BGP RIB for Test Case Three 73

Figure 4.19: BGP Sample Update for Test Case Three...76

Figure 4.20: Database Table for Test Case Three 76

Figure 4.21: Output of Test Case Three 77

Figure 4.22: Topology for Test Case Four 79

Figure 4.23: BGP RIB of Test Case Four - Non-determinism 80

Figure 4.24: BGP Sample Updates of Test Case Four 81

Figure 4.25: Database Table for Test Case Four 81

Figure 4.26: Program Output of Test Case Four 82

Figure 4.27 : Our Test Network with CCS 83

Figure 4.28: Sample BGP RIB for Real BGP Data 87

Figure 4.29: Sample BGP Updates for Real Data 88

Figure 4.30: Sample BGP Database Table 89

Figure 4.31: Sample Output of Real Data 90

Figure A.1: BGP Speaker of Test Network and Solana ND Connection Setup 94

Figure A.2: Network Configuration of Solana ND and Red Hat Linux Server 95

Figure A.3: BGP Data extraction from Solana ND 95

Figure A.4: Network Setup Among Windows Workstation, Solana ND and Red Hat Linux Server 96

Figure A.5: NetBeans Java Development Environment 97

Figure A.6: NetBeans Java Development Environment After opening The fault analysis software 97

Figure A.7: BGP fault analysis software 98

Figure A.8: BGP RIB and Updates Path Selection 99

Figure A.9: Output of Route Oscillation 100

List of Abbreviation

AS	Autonomous Systems
ASBR	Autonomous System Boundary Router
BGP	Border Gateway Protocol
CCS	Computing and Communication Service
EBGP	External Border Gateway Protocol
FSM	Finite State Machine
IBGP	Internal Border Gateway Protocol
ICMP	Internet Control Message Protocol
IDE	Integrated Development Environment
IGMP	Internet Group Management Protocol
IOS	Internetwork Operating System
IP	Internet Protocol
ISP	Internet Service Provider
MED	Multi Exit Discriminator
MRAI	Minimum Route Advertisement Interval
ND	Network Device
NLRI	Network Layer Reachability Information
OSPF	Open Shortest Path First
RIB	Routing Information Base
SPP	Stable Path Problem
SPVP	Simple Path Vector Protocol
TCP	Transport Control Protocol
WSC	Web Service Client
XML	Extensible Markup Language

Chapter One

Introduction

1.1 Introduction

Maintaining good performance of a backbone network needs automated monitoring of the network to identify problems in timely manner, generate a quick response, and guide the system administrator to contain the ill effects of failures in the end systems. Network administrators need to know when destination networks become unreachable so that they can notify affected customers and launch a process of identifying the cause of the problem. Once the problem is identified then the operators may respond by taking necessary actions for solving the problems. But these kinds of measurements are very crucial in an existing network protocols, router implementations, and operational practices to drive enhancements for the future. All of these tasks require effective ways to work on huge amounts of real time data to produce concise and significant reports about necessary modifications in network conditions.

The Border Gateway Protocol (BGP) is a path-vector protocol, which is identical to a distance vector protocol with additional path information. Routing information in BGP contains multiple attributes, which are used in route selection and distribution in the network. Keeping the length of the path to the destination and the list of ASes that form the path to the destination eliminates the “routing loop problem” among the ASes, which is a well-known problem associated with distance vector protocols. The iBGP protocol is also used to prevent routing loop inside an AS. The simplicity in specification is mainly due to the enormous flexibility the protocol offers to the operators for configuring specific policies to select the best paths in the BGP decision process.

The performance of a backbone network is especially vulnerable to *inter-domain* routing changes that affect how data traffic travels to destinations in other Autonomous Systems (ASes). The ASes use the Border Gateway Protocol (BGP) which is the current inter-domain routing protocol, to exchange information among them and to know how to reach destination address blocks or prefixes in the Internet. BGP is a policy-based routing that allows each AS to independently define a set of local policies concerning which routes it imports, which route is the best when more than one route becomes available for a single prefix and which routes it advertises to other networks. However, since routes are selected based on local policies of the ASes, it might cause global conflicts or network topology disputes among the ASes, such as route oscillation or non-determinism. The route oscillation problems becomes increasingly crucial because of the exponential growth of the number of ASes in the Internet. According to the statistics of World autonomous system number currently the total number of autonomous systems is around 60,200 which was very low even up to few years ago.

The main goal of this thesis is to perform extensive study, in-depth analysis and develop a solution to mitigate the existing problems. In this thesis, first we present a survey and a thorough analysis of the existing available solutions and their implementations in dealing with the BGP policy induced faults, which are proposed by the researchers and commercial communities working on the route oscillation problem. Then, we proposed and explained the development of an algorithm for identifying the BGP topology disputes or route instability within an autonomous system due to local policy. To this end, real time BGP routes and updates are collected continuously through Solana Network Device as XML files. Furthermore, Java programming language is used to implement the algorithm, and these XML files are used as input for the proposed tool. The reports should focus on routing changes that disrupt reachability, route instability because of some local BGP policies, generate a large number of update messages and affect a large volume of traffic as well. There may be some similarities with some recent work on BGP root-cause analysis of routing changes [6,7,15, 18] however our approach differs noticeably from these works. We have evaluated our system on two types of data, one is real time data collected by setting up an Autonomous System and establishing EBGP connectivity with Ryerson University Autonomous System. We designed four test cases using GNS3 simulator and collect data from these virtual backbone networks. Our system analyzes about 400 thousands of BGP routes, receives BGP update messages every 15 minutes and BGP data from virtual backbone networks. Hence, in our approach we try to identify the problems and produce reports that serve as immediate needsfor the network operators.

Finally but most significantly, we proposed a new solution in the thesis that automatically detects and eliminates the instability in the best path selection process due to local policy conflicts among ASes. Previous solutions require expensive path histories with high overhead that also prevent ASes from locally choosing their routing policy. Our proposed solution reduces the overhead by giving the ASes flexibility in of choosing any routing policy. Based on the detection of route instability each AS dynamically adjusts its own path preference by moving to the immediate next criteria or attribute for selecting the best path. We also present the test cases that we developed using BGP simulator for simulating oscillation faults in the test-bed and then discuss their results.

1.2 Thesis Overview

This thesis is organized as follows. In Chapter 2, we first give background of BGP and then summarize the related work in this area to point out the difference from our work. Based on the methodologies of the related works the main areas of our study are, (a) The analysis of BGP convergence through simulations, (b) BGP protocol analysis to know policy divergence and configuration to prevent route selection anomalies and finally (c) Analysis of BGP Topology disputes due to different policies and some proposed algorithm to overcome this disputes.

In chapter 3 we define the problem and present in-depth analysis of BGP Policy induced route anomalies specially route oscillation and non-determinism. Our proposed solution to resolve Inter-Domain policy conflicts due to BGP local policy is also described in this chapter. The validation and output of the algorithm is also shown in this chapter.

Chapter 4 presents our proposed BGP fault detection algorithm and the software design specially to detect oscillation and non-determinism within an AS. This chapter also describes our test network, test cases developed by using simulator for the purposes of policy induced fault detection, and results based on our proposed fault detection tool to detect route instability within an AS.

Finally chapter 5 provides conclusion and future works.

Chapter Two

Background and Literature Review

2.1 Internet Architecture

The vast Internet network has been formed by the progressive and incremental interconnection of a huge number of networks, each include a very large number of links, end systems, and intermediate systems. Companies rely upon this communication infrastructure for doing their business and it also influences the daily lives of numerous number of people. The Internet system consists of a number of interconnected packet networks supporting communication among host computers using the Internet protocols. These protocols include the Internet Protocol (IP), the Internet Control Message Protocol (ICMP), the Internet Group Management Protocol (IGMP), and a variety transport and application protocols that depend upon them. All Internet protocols use IP as the basic data transport mechanism. IP is a datagram, or connectionless, internetwork service and includes provision for addressing, type-of-service specification, fragmentation and reassembly, and security. ICMP and IGMP are considered integral parts of IP, although they are architecturally layered upon IP. Reliable data delivery is provided in the Internet protocol suite by Transport Layer protocols such as the Transmission Control Protocol (TCP), which provides end-end retransmission, resequencing and connection control. To communicate using the Internet system, a host must implement the layered set of protocols comprising the Internet protocol suite [15].

In the Internet model, constituent networks are connected together by IP datagram forwarders which are called routers or IP routers. A router connects to two or more logical interfaces, represented by IP subnets or unnumbered point to point lines. Thus, it has at least one physical interface. Forwarding an IP datagram generally requires the router to choose the address and relevant interface of the next-hop router or (for the final hop) the destination host. This choice, called relaying or forwarding depends upon a route database within the router. The route database is also called a routing table or forwarding table. The term "router" derives from the process of building this route database; routing protocols and configuration interact in a process called routing. The routing database should be maintained dynamically to reflect the current topology of the Internet system. A router normally accomplishes this by participating in distributed routing and reachability algorithms with other routers [15].

An Autonomous System (AS) is a connected segment of a network topology that consists of a collection of subnetworks (with hosts attached) interconnected by a set of routers. The subnetworks and the routers are expected to be under the control of a single operations and maintenance (O&M) organization. Within an AS routers may use one or more interior routing protocols, and sometimes several sets of metrics. An AS is expected to present to other ASes an appearance of a coherent interior routing plan, and a consistent picture of the destinations reachable through the AS. An AS is identified by an Autonomous System number [15]. Internet is in fact the composition of this Autonomous Systems (AS) to manage the complexity, each AS is in independent administrative control.

ASes join the Internet for different purposes. The enormous number of the ASes are simply interested in getting access to the network, e.g., in order to access the content that is provided over the Internet. These domains are called stub domains. Some ASes makes business by providing access to the Internet, that is, by making certain that packets can traverse the network and reach their destination. Such enterprises called Internet Service Providers (ISPs), sell the service of transiting packets across their own network infrastructure, and are therefore called transit ASes. A stub AS purchases connectivity to an ISP by paying a fixed amount of money for each bit of information that is sent across the network. Transit ASes can interconnect their networks using a number of policies. ISPs have three common relationships:

- *Customer- Provider:* Where one ISP pays another to forward its traffic.
- *Peer-Peer:* Where two ISPs agree that connecting directly to each other (typically without exchanging payment) would mutually benefit both, perhaps because roughly equal amounts of traffic flow between their networks.
- *Backup:* Where two ISPs set up a link between them that is to be use only in the event that the primary routes become unavailable due to failure [5].

From routing's point of view, Each AS constitutes a distinct routing domain which normally also belongs to a single administration domain. Routers communicate with each other using one or multiple intra-domain routing protocols within an AS, such as OSPF, EIGRP and RIP. ASes are connected via routers which exchange information through inter-domain routing protocol. The most popular inter-domain routing protocol currently in use is Border Gateway Protocol (BGP).

2.2 BGP Overview

Internet routing is destination-based and this destinations are expressed as network prefixes. The Border Gateway Protocol (BGP) is the routing protocol that ASes use to exchange information about how to reach destination address blocks (or *prefixes*)[10]. BGP has two different modes of operation:

EBGP (External BGP) exchanges reachability information between Autonomous Systems. The border routers of the network learn how to reach external destinations through eBGP sessions with routers in other ASes. A large network often has multiple eBGP sessions with another AS at different routers. This is a common requirement for two ASes to have a peering relationship, and even some customers connect in multiple locations for enhanced reliability. Figure 2.1 shows AS 100 has two EBGP sessions with AS 200 and one EBGP sessions with AS 300.

IBGP (Internal BGP) exchanges external reachability information within an Autonomous System. IBGP allows internal sources to reach external destinations but it does not maintain reachability within an AS. This is done by IGPs such as EIGRP, OSPF or RIP. By applying local policies to the learned routes, a border router selects a single best route and uses IBGP to advertise the route to other border routers within the AS. In the simplest case, each router has an IBGP session with every other router (*i.e.,* a *full-mesh* iBGP configuration). Figure 2.1 shows the router R6, R7 and R8 has IBGP sessions within the AS 200.

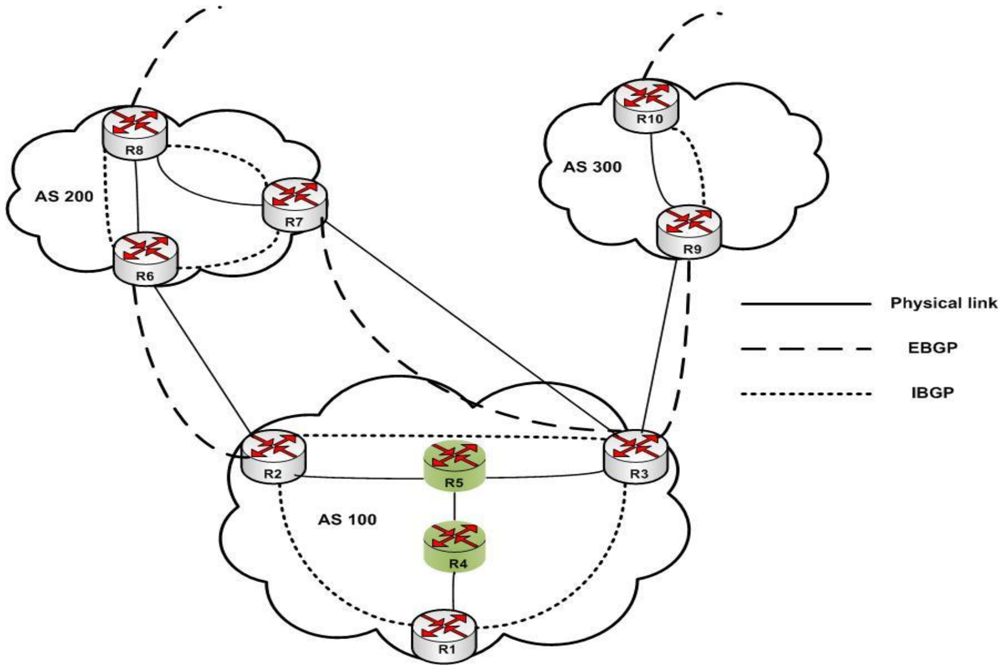


Figure 2.1: General Routing Architecture

BGP uses TCP as its transport protocol to ensure reliable delivery. The connection between a pair of BGP routers is called a BGP session and participating routers in a BGP session are called peers (or BGP speakers). After a BGP session is established, its peers will exchange the information about entire routing table. Afterwards only incremental updates to the routing table will be exchanged. There are four types of BGP messages exchanged between two speakers:

Open is used to start a BGP session, each BGP peer must send each of its neighboring peers an OPEN message. The OPEN message contains information about the local BGP speaker, and is used after a TCP session has been established. All fields in the OPEN message must be negotiated and accepted before a session can exchange routing information. For an example in figure 2.1 router 2 and Router 6 use BGP OPEN messages to form a BGP session. Router 2 sends an OPEN message to Router 6 containing its BGP version of 4, the My AS value of 100, a hold timer of 180 seconds, and the BGP ID. Router 6 responds with its own OPEN message containing its local BGP version of 4, it My AS of 200, a hold timer of 180 seconds, and the BGP ID. In this example, both of the BGP speakers are in a different AS and will participate in an E-BGP session.

Update message exchange routing information between peering routers, after a BGP session has been established. It contains information about each route advertised to the peering router. The information contained in BGP UPDATE messages are Unfeasible Route Length, Withdrawn Routes, Total Path Attribute Length, Path Attributes and Network Layer Reachability Information (NLRI). There are two types of update messages that BGP protocol use: announcements and withdrawals

An ***announcement*** message sent from a router R_2 in AS 100 to a router R_6 in AS 200 to the destination prefix d_P with routing information R indicates that R_2 is willing to carry traffic from R_6 destined to d_P following the route R. The routing information advertised also indicates the routing decisions of the advertiser. In this case, R_2 uses the route R to reach destination d_P. Given all the available routes received from one's neighbors stored in the RIB (Routing Information Base), the router uses its routing policies to select the best route. In this example, R_6 may have received multiple routes for destination d_P, one from each of its providers. It will choose the best one based on its own routing policies and may in turn announce the route to its customers. Routing policies also determine the available routes between a pair of Internet hosts. R_2 may decide not to advertise the route to R_6 due to its specific policies. Because of the way in which BGP policies filter and propagate routes, physical connectivity on the Internet does not imply logical reachability. Thus, BGP does not ensure global reachability, which is important for assure the ability for any to any communication.

Withdrawals, as expected, indicate that the sender no longer has a route to reach the given destination. They invalidate the last announcement sent regarding the route. This can be caused by things such as transient failures caused by a flapping link, congestion which can cause session time out, and software upgrade on routers which may require reboot. Announcements can also serve as implicit withdrawals, modifying the attributes of the route that was last sent.

Notification messages are used to indicate an error condition resulting in BGP session termination. It is always immediately followed by session termination. Upon termination of a BGP connection, the TCP session between the BGP peers is torn down, all resources are released, "route withdrawal" messages are sent to peering BGP peers, and all BGP routes are removed from the table.

Keepalive messages are sent by the peering routers to notify neighboring peers that the BGP connection is active. The message interval can be changed from the default value (60 seconds) to any other value between 3 and 4,294,967,295. This timers might also be set to 1/3 the negotiated hold-timer value, which is, by default, 180 seconds.

BGP peers transition through several states before becoming adjacent neighbors and exchanging routing information. During each of the states, the peers must send and receive messages, process message data, and initialize resources before proceeding to the next state. This process is known as the BGP *Finite-State Machine (FSM)*. If the process fails at any point, the session is torn down and the peers both transition back to an Idle state and begin the process again. Each time a session is torn down, all routes from the peer who is not up will be removed from the tables, which causes downtime. If configuration issues exist on one of the BGP peers, the peering routers continuously transition between unestablished states until the issue has been resolved. BGP peers transition through all the following states until an established BGP session has been created:

- Idle
- Connect

- Active
- Open Sent
- Open Confirmed
- Established

Each of these states also has accompanying *input events* (IEs). Input events are events occurring during a BGP session that trigger an action.

BGP's routing information base consists of three parts: Adj-RIBs-In, Loc-RIB, Adj-RIBs-Out. The Adj-RIBs-In contains unprocessed routing information that has been advertised by its peers. The Loc-RIB contains the routes that have been selected by the local BGP speaker's decision process according to certain local policies. And the Adj-RIBs-Out organizes the routes for advertising to specific peers through the local speaker's UPDATE messages [3]. During this procedure, BGP could enforce its own policies by selecting the incoming and outgoing routing information. Therefore, BGP doesn't simply obey the shortest-path selection criteria.

2.3 BGP Attributes

Routes learned via BGP have associated attributes that are used to determine the best route to a destination when multiple paths exist to a particular destination. These properties are referred to as BGP attributes, while BGP allows the network operators to manipulate these attributes. These attributes can be categorise as follows:

- *Well-know mandatory* attributes must be present in the BGP update message and must be implemented by all BGP speakers.
- *Well-know discretionary* attributes recognized by all BGP implementations but it may or may not be in BGP update packets.
- *Optional* attributes may not be recognized by a BGP implementation.
- *Non-transitive* attributes will removed if it is not recognized by a BGP speaker and will be passed to the peer if it is recognized by a BGP speaker.
- *Transitive* attributes will be passed to the BGP peer, even if it is not recognized.

- *Complete* attribute is recognized by all downstream routers.
- *Partial* attribute is not recognized by all downstream routers.

We will try to explain the attributes in the following section :

Origin is a Well-known mandatory attribute and specifics how a prefix is learned or the origin of the route. This attribute can assume three values:

- IGP indicates the route is internal to the originating AS. This route is defined by BGP Network command or when IGP is redistributed into BGP.
- EGP indicates route or NLRI is learned via EGP.
- INCOMPLETE indicates that the route is learned through something other than an IGP or EGP. This usually learned when a static route redistributed into BGP.

Next-Hop attribute is also a well-known mandatory attribute. Next hop is the IP address of the EBGP peer that advertises the route to other AS. IBGP advertisement does not change the next hop address. The router makes a recursive lookup to find the BGP next hop in the routing table. The advertise route is only feasible if the router that receives the route advertisement knows how to reach the next hop.

AS-Path is a well-known mandatory attribute. It is simply a sequence of AS numbers that represents the path has traversed in order to reach a destination. BGP uses this attribute to detect a routing loop and policy decision, shorter AS-Path length is more preferable. Routing policy can also be implemented by manipulating the AS-Path length. A BGP speaker advertises only the route that it uses to its peer or neighbour routers. Upon received of an updates each router evaluates the path vector and invalidates any route which includes the own AS number in the AS path. After this test a path may be rejected or accepted depending on the import policy.

Weight is a Cisco proprietary attribute and considered as first attribute or the highest precedence for BGP best path selection. It is assigned locally to a BGP router and not propagated to other BGP router. The default value of weight is 0 and the range is from 0 to 65535. The route with maximum weight value will be considered as the best route. By default learned routes have weight 0 and self originated routes has weight 32768.

Local Preference attribute is a well-know discretionary attribute that indicates the degree of preference on a route. This attribute is only passed among IBGP peer that this is only significant within an AS. This value plays a very vital role in the route selection process that the route with a higher degree of preference is selected as best path. It influence the exit point or egress traffic from a local autonomous system. A higher value means a higher preference and the default value is 100. Local preference is set via the "bgp default local-preference" command or with route-maps.

Multi-Exit-Disc (MED) attribute is an optional non-transitive attribute. It provides a hint to the external neighbours about the preferred path into an AS. It influence another AS on which way to choose in order to reach a certain route given that we have multiple entry points into that AS, that is it influences ingress traffic. This attribute exchanged between EBGP peers or ASes. A lower MED value means a higher preference and the range of the values are from 0 to 65535.

Community is an optional transitive attribute to identify a group of destinations that share some common properties. Each autonomous system administrator may define which communities a destination belongs to. By default, all destinations belong to the general Internet community [5]. It offers a technique to logically classify prefixes for use in policies by attaching an identifier that is significant within a network. Prefixes can have multiple communities and policy decisions can be made on one or more of them. Route-map is used to set the community attributes.

Some well-known communities are

- No-Export: A BGP route receives with this community value should not advertise to the external BGP peers.
- No-Advertise: A BGP routes receives with this community value should not advertise to any BGP peers.
- Internet: A BGP routes receives with this community value should advertise to anyone in the internet.

Atomic Aggregate attribute is a well-known discretionary attribute. When a BGP router aggregates a set of prefixes from different sources this attribute is set to indicate the information loss. When a BGP speaker receives a route with this attribute -

- Needs to be aware of the fact that the actual path to the destination, as specified in the NLRI of the route may traverse ASes that are not listed in the AS-Path attribute.
- Should not eliminate the attribute from the route when advertising it to other speakers.
- should not make any NLRI of the route more specific when advertising route to other BGP speaker.

Aggregator is an optional transitive attribute and specify the BGP router that performs the aggregation. In addition this attribute indicates which router originated the aggregate.

2.4 Routing Policy and Decision Process

A BGP speaking router may learn multiple paths to the same destination from its internal and external neighbors. The decision about which paths to accept from a neighbour and to decide which set of routes should be advertised to each peer is based on routing policy. Policies are set of rules that are configured locally by network operators and are not shared with any ASes. Being a policy based routing protocol, BGP gives ASes the freedom of applying their own routing policies to select the best routes, and to propagate their best routes to the selected peers only. Policies are also used to enforced business agreements made between two or more ASes.

A BGP router select the best route for each destination from a set of routes according to the following criteria:

1. Is the route valid ? A route is valid if:
    - The route must be synchronized with the Interior Gateway Protocol if BGP synchronization is enabled.
    - The NEXT_HOP is reachable.
    - The AS_PATHs received from an external AS must not contain the local AS Number.
    - The local routing policy permit the route.

2. The first valid path received is automatically the 'best path'. Any further paths received are compared to this path to determine if the new path is 'best'.

3. Select the highest Weight path. Weight is a Cisco-proprietary attribute and is local to a BGP router on which it is configured.

4. Choose the path with highest Local Preference value (AS label significant and default value is 100).

5. Prefer the route locally originated through Network command, Aggregation or Redistribution over received routes.

6. Select the path with shortest AS-PATH length.

7. Select the path with lowest origin type:
   IGP < EGP < INCOMPLETE

8. Smallest Multi-Exit-Discriminator (MED) value is preferred. This comparison only occurs if the first AS number in the AS path is the same for multiple paths.

9. Prefer eBGP learned route over iBGP learned route

10. The path with the smallest IGP metric to the exit point is preferred.

11. Prefer first received external route if more than one path are external (the oldest external path).

12. Prefer the route that receives from the BGP router with lowest router ID. The administrator can manually define the Router ID using command otherwise router ID will be the highest IP address on the router.

13. Minimum Cluster_ID length will select next if the router ID is the same for multiple routes.

14. Select the path that comes from the smallest neighbor address

From this list, we can see it is impossible for two BGP routes to tie each other and become equally preferred. As has been stated elsewhere in this thesis, BGP contains only a single best path to any given destination. BGP runs across the entire internet, therefore it must manage to reduce the number of advertised routes in order to prevent the internet from becoming flooded with route advertisement traffic. Thus, this algorithm is used to select only one best path to a destination. Figure 2.2 shows the complete process how BGP select its best route.

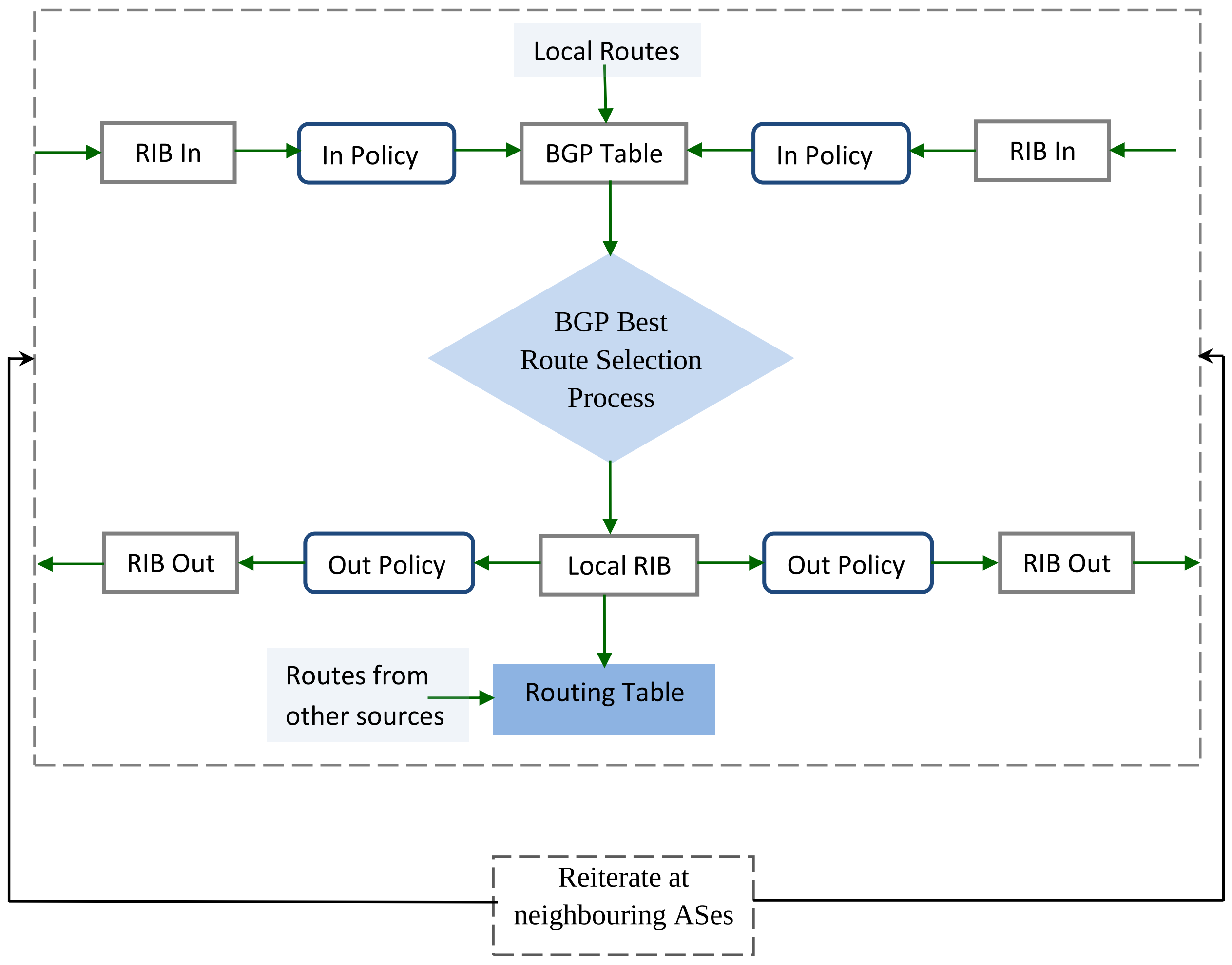


Figure 2.2: BGP Route Selection Method

A significant problem of defining routing policies independently within the ASes is that the policies defined locally can lead to BGP protocol route instability that never reach a stable routing. Such protocol divergence and proposed solutions has been illustrated by many research. In this study we also done an extensive analysis concerning inter-domain policy disputes and propose two solutions. In this thesis we first explain our propose algorithm so that BGP protocol itself can identify route instability specially oscillation and automatically resolve this instability due to policy conflict. After that it give details of our proposed tool to detect route instability within an AS and report that to network administrator so that administrator can take action to solve that promptly.

2.5 Related Works

In Griffin *et al.* [21], it has shown that policies configured independently by ASes can cause route instability at the network-wide level and never converge to a stable routing. Unfortunately, even if the policies were fully known, the complexity for statically checking policy convergence is NP-hard. Subsequent work by Griffin *et al.* [20, 21] uses formal analysis to model BGP path selection based on *Stable Path Problem (SPP)* and after that *Simple Path Vector Protocol (SPVP)* that abstracts away all non-essential details. SPVP is a distributed algorithm to solve Stable Path Problem. Ee et al. [14] has proposed a solution for policy induced route instability based on *Precedence Metric*. Cobb, J. A. and Musunuri, R. [11, 12] also proposed divergence detection protocol (DDP) and bounded divergence protocol (BDP) to enforce convergence in Inter-Domain policy conflicts. The use of MED attribute is another reason for route oscillation shown by many researchers [2, 22, 25]. More recently, some researchers also proposed some solutions and internet draft [27, 32, 33, 34] for this MED-induced route oscillation. Some very useful BGP monitoring tools also proposed in previous researches [10, 35]. The work discussed in this thesis also benefits from such formal analysis and the experimental measurement to recognize routing anomalies.

Chapter Three

Policy Induced Topology Disputes and Proposed Solutions

This Chapter explores BGP topology disputes due to local policies of individual ASes. A networked system often build up by a collection of mutually dependent subsystems sharing resources and administered by locally determined various policies. Generally, a subsystem not only possesses local resources but also based on local preference to rely on other subsystems for additional resources. It is very crucial for network systems to maintain interactions among inter-dependent subsystems and use local preference. In this chapter we study and analyse the adverse consequences arising in these interactions, such as oscillations and non-determinism. We define oscillation and non-determinism as a networked system fails to settle down at the best route for a particular destination and system may exhibits unpredictable behaviour in best route selection respectively.

The Border Gateway Protocol (BGP) is a widely-used protocol for policy-based routing. Its rich semantics allows unrestricted and diverse routing policies to be specified by routing systems. There are many attributes for specifying the preferences among paths in BGP, such as AS path length, Local Preference, MED, IGP cost, Community tags etc. The volume of the routing update messages can be very large and each message only shows AS-path changes but not why and where they happened. Changing any of the attributes on BGP router could influence the best route selection process from the local BGP neighbors within the same AS and the external BGP neighbors within other ASes. Even though BGP itself has been well studied, there are still remaining oscillation and non-deterministic problems in inter-domain routing due to certain settings of these attributes among the ASes. In this thesis we will present a comprehensive study of their properties and consequences in the context of networked systems, taking into account the structures of dependence and preferences of subsystems.

Previous works have already been described some of these policy-induced routing problems. A simple instance of BGP policy-induced route oscillation is known as oscillation gadget [21]. Another simple setting in policy-based routing that can cause unintended non-determinism termed as BGP wedgies [19]. Multi-Exit Discriminator (MED), a numerical value specified by a sink router to rank the multi-homing paths to it. This MED-Induced oscillation is a very well known problem in BGP [2,23,28] but there is a draft solution [34] for this problem. We next present several instances of BGP polices-induced routing oscillation. Our main focus is to identify the BGP route instability within the AS so that it can inform the network administrator as early as possible and propose a solution for BGP protocol to resolve this inter-domain policy disputes automatically.

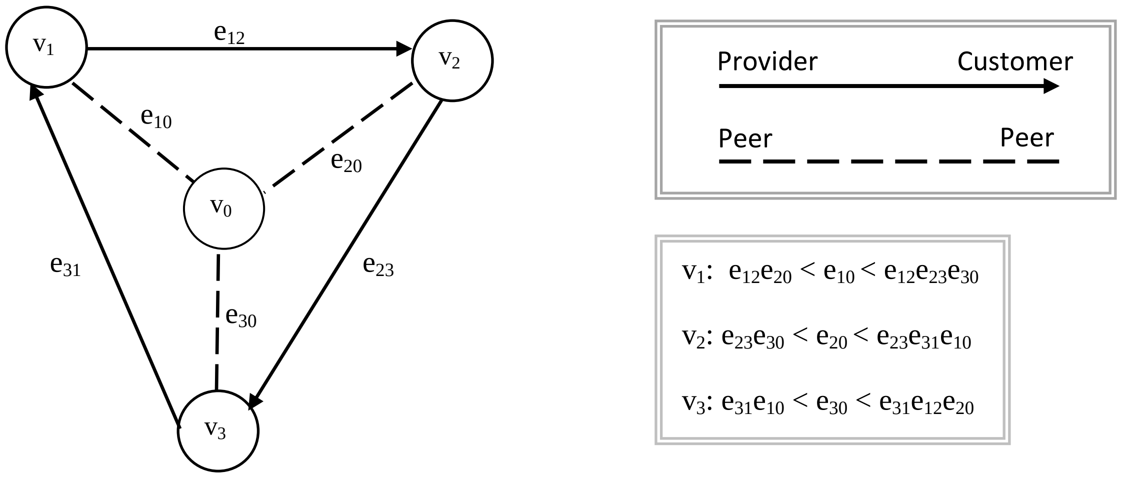
3.1 BGP Policy Disputes

3.1.1 BGP Policy Induced Oscillation

Routing policies for EBGP usually define locally and consider business relationship among the ASes as:

- *Provider-Customer*: Providers are liable to forward traffic for their customers.
- *Peer-Peer*: ASes peer with each other to facilitate mutual traffic flow.
- *Backup:* Where two ASes set up a link between them that is to be use only in the event that the primary routes become unavailable due to failure

A common practice is that a router prefers to forward traffic via a customer to those via a peer or a provider, subject to avoiding long paths of certain length. The configurations of forwarding paths should satisfy the local consistency of dependence. So in inter-domain routing, we observe a policy-induced oscillation in a simple setting called the bad triangle or bad gadget [21]. Consider Figure 3.1, where the network is represented as the AS graph G = (V, E), each node $v_i \in V$ corresponds to an autonomous system, where i={0,1,2,.........,n} and each edge $e_{ij} \in E$ corresponds to a BGP session between ASes v_i to $v_{j,}$, where i={0,1,2,.........,n} and j={0,1,2,.........,n} but $i \neq j$. We assume that node v_0 is the origin, it is the destination to which all other nodes attempt to establish a path but local preferences of their paths to destination v_0 may different. A path *p* is an edge or a sequence of edges or sequence of nodes, because each edge is actually a sequence of two connected nodes $e_{ij} = (v_i\ v_j)$. So a path *p* in G is a sequence of connected nodes, $(v_i\ v_j \ldots \ldots \ldots v_0)$ where i, j > 0 and $i \neq j$. For each $v \in V$, p^v denotes the set of permitted paths from v_i to the origin $v_{0.}$ If $p_1 = (v_i\ v_j \ldots \ldots \ldots v_0)$ is in p^v the node v_j is the next hop AS. If path p_1 is preferable to path p_2, we rank as: $r(p_1) < r(p_2)$ where r is a rank function. So it may creates a dispute in the rankings among routing systems, between subpaths and superpaths (e.g. $e_{23}e_{30} < e_{20}$, whereas $e_{12}e_{20} < e_{12}e_{23}e_{30}$).



v_1
v_2
v_0
v_3
e_{12}
e_{10}
e_{20}
e_{31}
e_{23}
e_{30}
Provider
Customer
Peer
Peer
v_1: $e_{12}e_{20} < e_{10} < e_{12}e_{23}e_{30}$
v_2: $e_{23}e_{30} < e_{20} < e_{23}e_{31}e_{10}$
v_3: $e_{31}e_{10} < e_{30} < e_{31}e_{12}e_{20}$

Figure 3.1: A triangular network and Local policy

Considering the Figure 3.1. The business relations will induce the following rankings:

$v_1 : e_{12}e_{20} < e_{10} < e_{12}e_{23}e_{30}$, $v_2 : e_{23}e_{30} < e_{20} < e_{23}e_{31}e_{10}$, $v_3 : e_{31}e_{10} < e_{30} < e_{31}e_{12}e_{20}$.

For instance, v_1 ranks as $e_{12}e_{20} < e_{10}$ because the path traversing customer v_2 is preferable to the one via a peer, and $e_{10} < e_{12}e_{23}e_{30}$ because the long paths of length greater than 2 are avoided. This forms a cyclic structure as:

$e_{12}e_{20} < e_{10}$ [*as a subpath of*] $e_{31}e_{10} < e_{30}$ [*as a subpath of*] $e_{23}e_{30} < e_{20}$ [*as a subpath of*] $e_{12}e_{20}$

This therefore generates an oscillation.

Example 3.2 displays a cyclic structure that causes an oscillation. In this example we try to analyze the behaviour of the topology node (AS) by node and start our observation from node v_3. After step seven it comes back to step two again and continue the cycle.

In this loop if we look every AS individually we see --

v_3 select route e_{30} and $e_{31}e_{10}$ alternatively
v_2 select route e_{20} and $e_{23}e_{30}$ alternatively
v_1 select route e_{10} and $e_{12}e_{20}$ alternatively

In the figure 3.2 we mark the selected route as grey color.

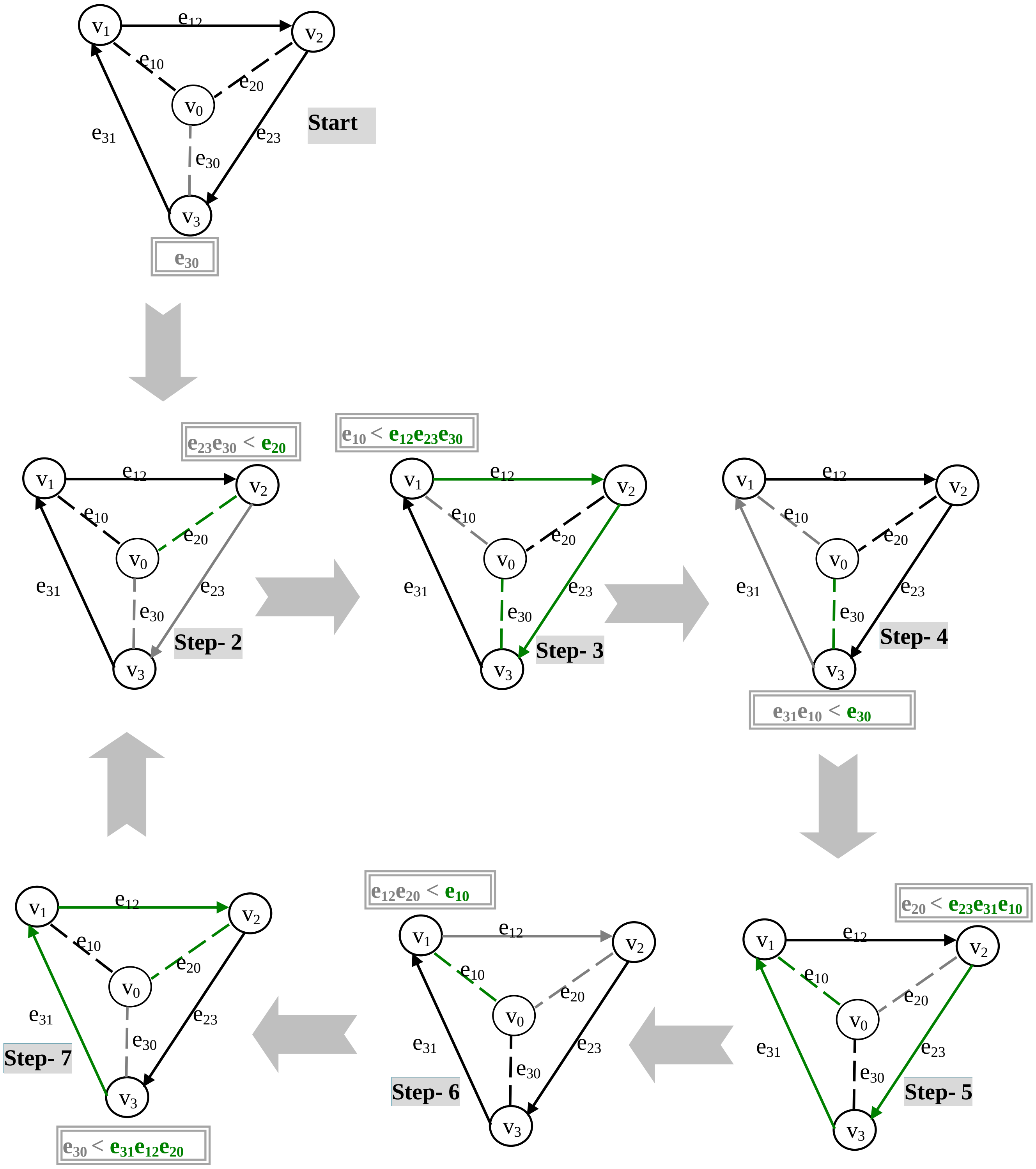


Figure 3.2: Node by Node Observation of Oscillating Gadget

In figure 3.3 we try to observe the oscillation of the whole topology at a time and same things happened in each nodes as figure 3.2.

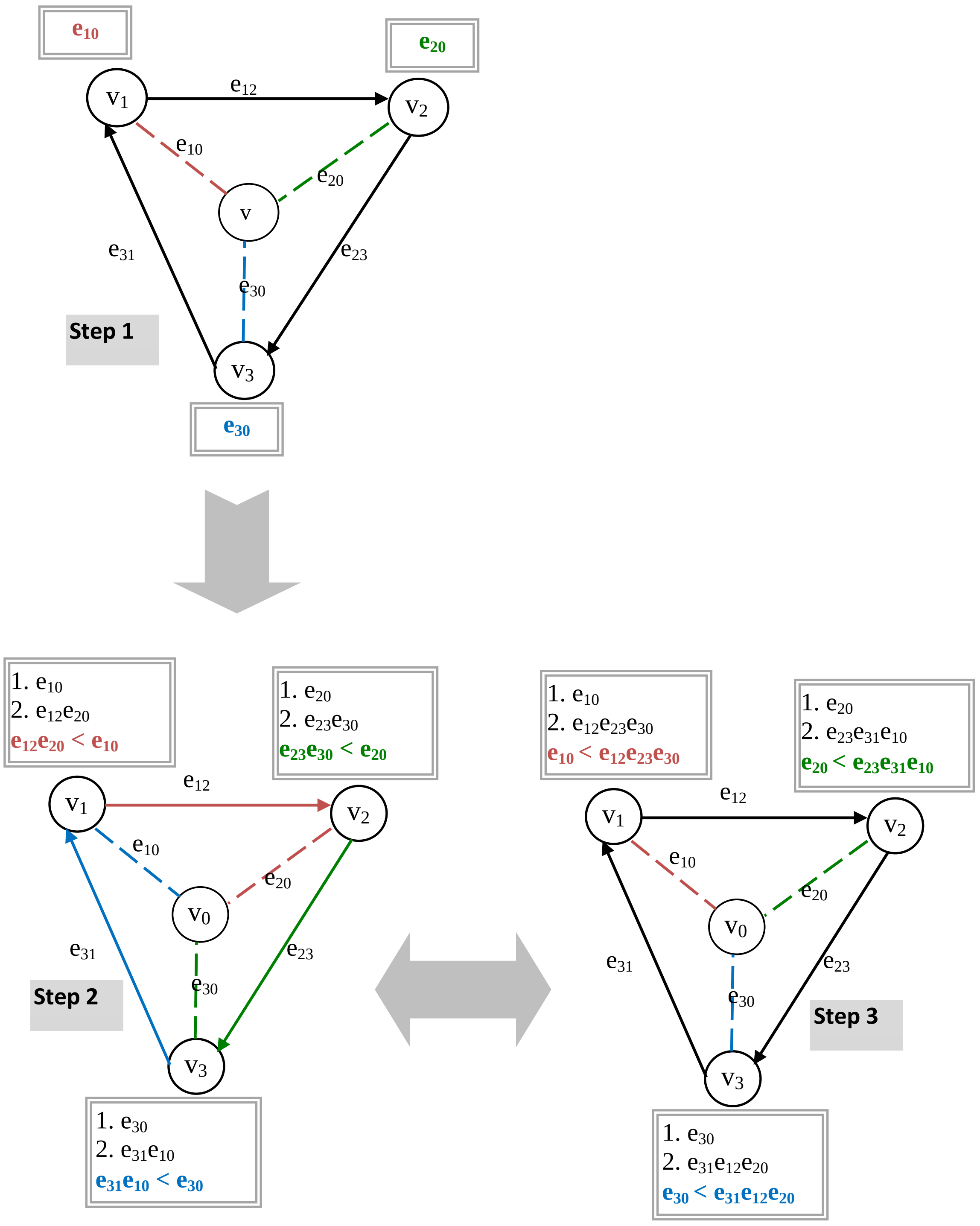


Figure 3.3: Observation of Oscillating Gadget whole topology

3.1.2 Dispute Wheel

Any of the such oscillation or disputed topology can be characterize by a new structure called *dispute wheel* [20, 21]. A dispute wheel consists set of nodes, v_0, v_1,v_2, ..., v_k and two types of paths Q_1, Q_2, ..., Q_k and R_1, R_2, ..., R_k. For each $0 < i \leq k$, dispute wheel can be define as follows :

- R_i is a path from v_i to v_{i+1} known as rim path.
- Q_i is a path from v_i to v_0 (destination) known as spoke path.
- A path $p_1 = R_iQ_{i+1}$ is a permitted path at v_i
- A path $p_2 = Q_i$ is less preferred than $p_1 = R_iQ_{i+1}$ at node v_i

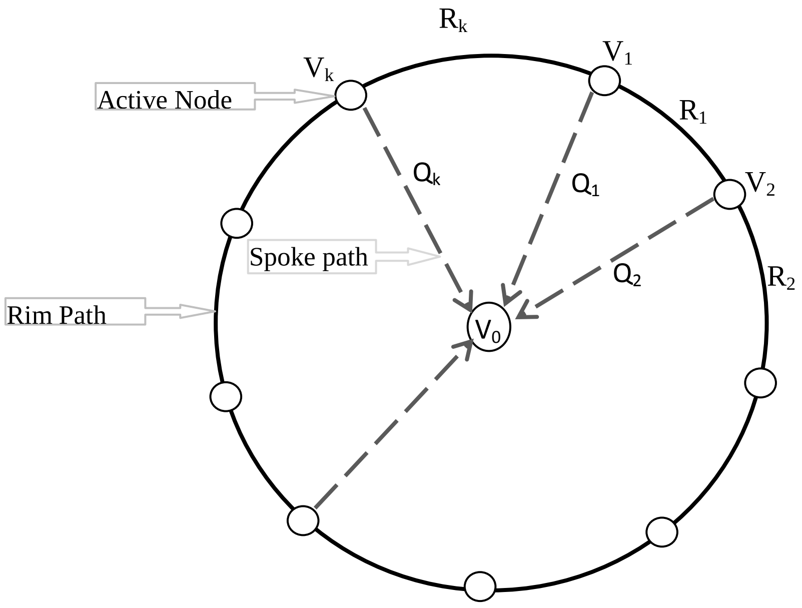


Figure 3.4: Dispute Wheel

A general dispute wheel is illustrated in figure 3.4. It comprises of rim paths and spoke paths. The node where the spoke path and rim path has connected is known as active node. So active node also has a direct spoke path. There may be some nodes which are only within the rim path and do not have any spoke path. These nodes are called rim node of inactive node.

3.1.3 Policy-Induced Route Non-Determinism

The route non-determinism behavior of a topology is where a network may select an unpredictable path to the destination network. The local network's default routing policy often reflects a local preference to prefer routes learned from a customer to routes learned from some form of peering to optimize cost [19]. These preferences may be expressed via a local preference configuration setting, where the local preference overrides the AS path length metric of the base BGP operation. In the inter-domain routing environment it is commonly the case that a service provider may enter into arrangements with two or more upstream transit providers, passing routes to all upstream providers, and receiving traffic from all sources. If one upstream path fails, the traffic will switch to other links or backup path, again the primary path is recovered the traffic should switch back. The intent of this backup connections will be used for traffic only for the duration of a failure in the primary connection. To establish this kind of settings the provider publishes a set of community values that allows the client to select the provider's local preference setting. The client can use a community to mark a route as "backup only" towards the backup provider, and "primary' to the primary provider, assuming both providers support community values with such semantics. In this case, the local preference overrides the AS path length metric, so that if the route is marked "backup only", the route will be selected only when there is no other source of the route.

A link can be configured as a backup by using appropriate community tags to tolerate link failure. That is, the link will not be used except when no other path to the destination is available. Figure 3.4 shows the BGP settings with backup links for BGP wedgies [19] that can cause unintended non-determinism.

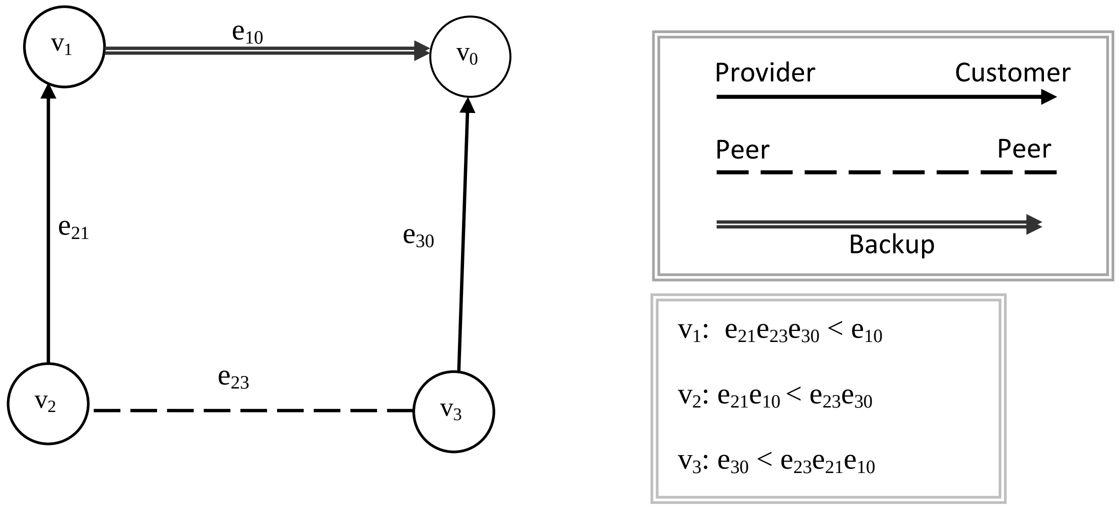

v_1
e_{10}
v_0
e_{21}
e_{30}
e_{23}
v_2
v_3
Provider
Customer
Peer
Peer
Backup
v_1: $e_{21}e_{23}e_{30} < e_{10}$
v_2: $e_{21}e_{10} < e_{23}e_{30}$
v_3: $e_{30} < e_{23}e_{21}e_{10}$

Figure 3.5: Network with business relation and backup link

Consider Figure 3.4, where e_{10} is a backup link, as specified by v_0 to v_1. Since routers may belong to different networks, community tags of an AS are normally not shared with others. Thus, AS v_2 and AS v_3 have no knowledge of e_{10} as it works as a backup link. The setting of business relations and the presence of a backup link will induce the following rankings:

v_1: $e_{21}e_{23}e_{30} < e_{10}$, v_2: $e_{21}e_{10} < e_{23}e_{30}$ and v_3: $e_{30} < e_{23}e_{21}e_{10}$

where v_1 prefers the non-backup link, while v_2 and v_3 prefers the customer path. This also forms a cyclic structure as:

$e_{21}e_{23}e_{30} < e_{10}$, and e_{10} is a sub path of $e_{21}e_{10}$.

Again $e_{21}e_{10} < e_{23}e_{30}$ and $e_{23}e_{30}$ *is a sub path of $e_{21}e_{23}e_{30}$*

So we can say $e_{21}e_{23}e_{30} < e_{10} < e_{21}e_{10} < e_{23}e_{30} < e_{21}e_{23}e_{30}$, which forms a cycle.

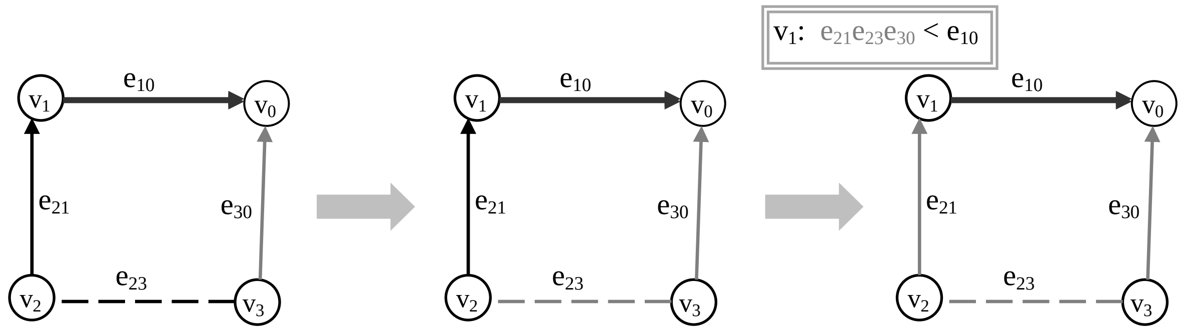


Figure 3.6: Network with intended path

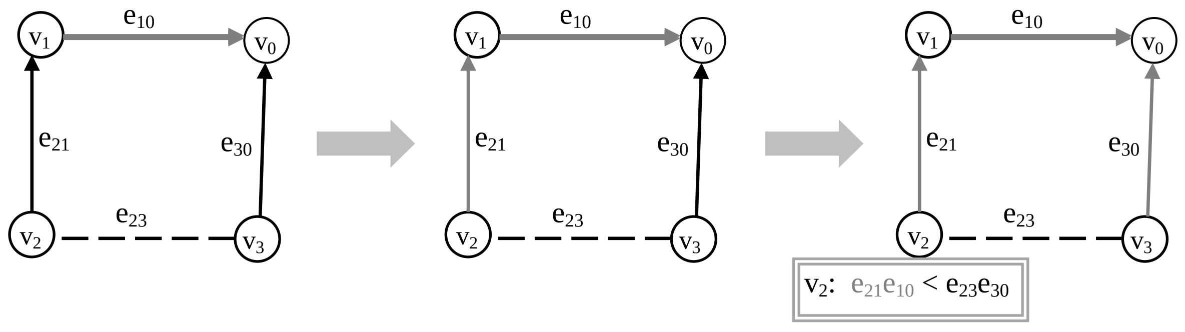


Figure 3.7: Network with unintended path

This therefore generates two settled forwarding configurations as in Figure 3.5 and 3.6. The intended configuration should use e_{30} rather than backup e_{10}. In figure 3.5 we show that node v_1 takes $e_{21}e_{23}e_{30}$ as its best path which is the intended situation. But when there is a reset or link failure of e_{30}, it will lead to an unintended configuration that retains e_{10}. In figure 3.6 node v_1 takes e_{10} as its best path and retain that even the intended path $e_{21}e_{23}e_{30}$ comes up again. The unintended configuration can only be reverted by resetting e_{10}. But resetting e_{10} is an adverse operation in an operating network. This also can create non-deterministic outcomes as the settled configuration may be eventually determined by some uncontrollable factors, such as resetting or failure of some links.

3.1.4 MED oscillation problem

The Multi-Exit-Discriminator (MED) attribute of a BGP route is a non-negative integer that is used to compare routes that pass through the same neighboring AS. The lower the MED value, the more preferred the route. MED comparisons only take place between routes which pass through the same neighboring AS. In such situations, the MED value of a route is used by the AS receiving traffic to indicate (to the sending AS) which links it prefers when receiving traffic. The BGP protocol requires that routers in the sending AS respect the MED values assigned to a route by the receiving AS[2]. Given current practices, we happen to see the problem manifest itself in the context of MED plus route reflectors or confederations[26].

The network in figure 3.7, AS100 is divided into two clusters. Cluster 1 has one route reflector (RR1) and two clients (C3 and C4). Cluster 2 has also one route reflector (RR2) and one client (C5). The route reflectors in AS100 are mesh networked and they advertise the best route from all received iBGP routes based on where that route came from. If the route comes from EBGP, the RR will forward it to all of its clients and the RRs in other clusters. If it is from a RR, it will forward to all clients in its cluster. And finally, if the route comes from a client, it will forward it to all RRs and its clients except the originator. In this example of MED oscillation, the automats systems AS10 and AS20 advertize a prefix originated from AS200 with unique MED values to AS100. Assuming at first that RR1 selects path P1 as it best route for the prefix. When RR1 receives the path P2 from C4, it will select it as its best route because of its lower IGP cost compared to P1 and advertize it to RR2. The route reflector RR2 will select path P3 as its best route because of its lower MED value compared to P2 since both paths (P2 and P3) came from the same AS. After that it will advertize it to RR1 who will select it as its best route for the same reason (lower MED value). After that RR1 will run its scanner and decide that P1 is its best route since it has a lower IGP cost compared to P3 and advertize it to RR2 which will select it for the same reason and withdraw its previous route (P3). After that RR1 will select P2 because it has a lower IGP cost compared to P1 and the loop continues. Now try to describe the oscillation process step by step as below.

1. Let at first route reflector RR1 select path P2 over path P1 (lower IGP metric) and Route Reflector RR2 select path P3 (only path known).
2. On receiving update from route reflector RR2, route reflector RR1 learns about path P3 and it selects path P1 as its best path (path P3 is ranked over path P2 based on lower MED value, and then path P1 is selected over path P3 based on the lower IGP metric)
3. Now on receiving the updates from route reflector RR1, route reflector RR2 learns about path P1 and it select path P1 as its best path over path P3 (lower IGP metric) and withdraws its previous best path P3.
4. When path p3 is withdrawn by route reflector RR2, route reflector RR1 select path P2 over P1 (lower IGP metric) and withdrawn its previous best path P1.
5. when path P1 is withdrawn by route reflector RR1, route reflector RR2 select path P3 over path P2 (lower MED value) and the cycle begin again.

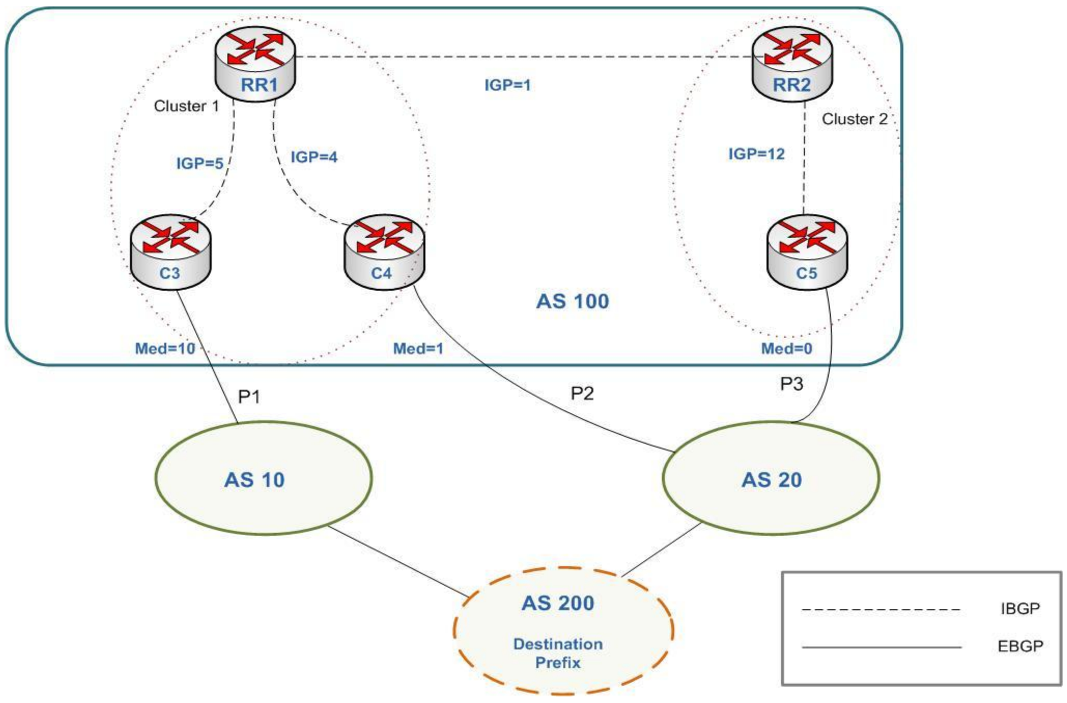


Figure 3.8: Oscillation due to MED

There is a solution developed by Walton et, al. (Internet Draft, December 15, 2011) [34] of this MED-Induced route oscillation. In order to eliminate the MED-induced route oscillations and to achieve consistent routing in a network, clearly a route reflector or a confederation ASBR needs to advertise more than just the best path for an address prefix. Two sets of paths for an address prefix that can be advertised by a BGP route reflector or confederation ASBR to eliminate the MED-induced route oscillations in a network. The first set involves all the available paths, and would achieve the same routing consistency as the full IBGP mesh. The second set, which is a subset of the first one, involves the neighbor-AS based Group Best Paths, and would be sufficient to eliminate the MED induced route oscillations.

3.2 Prior Solutions for Policy Induced Route Instability

3.2.1 Stable Path Problem (SPP)

To study the BGP policy based problems Griffin et al., 2002 [21]propose an abstract model of BGP through Stable Path Problem (SPP) and Simple Path Vector Protocol (SPVP). The Stable Path Problem consist of an undirected graph with a single destination. Each node in the graph has a set of permitted paths to the destination, which are routes learned from peers. A ranking function has also used to set an order of preference of the paths at each node, such that more preferable paths will have higher values assigned to them. A solution of an SPP is an assignment of permitted paths to the nodes that is consistent with the path chosen by its next-hop neighbor. Node u may choose the path P= (u,v,w,...,origin) only if the current path at node v is (v,w,...,origin) and path P is the current best path of node u.

The meaning of SPP can be summarized as follows: A network is represented as a simple, undirected, connected graph G = (V, E), where V = $\{V_0,V_1,...,V_n\}$ is the set of nodes connected by E, set of edges. Here nodes represent ASes and edges represent BGP sessions between ASes. For a node u, its set of peers is peers(u) = $\{ w| \{u,w\} \in E\}$. Node V_0 is the origin or the destination to which all other nodes are trying to set up a path. A path P in G is a sequence of nodes $(v_k, v_{k-1}, ..., v_1, v_0)$ such that $\{v_i,v_{i-1}\} \in E$ and for each i, $k \geq i \geq 1$. An empty path ϵ, indicates that a router cannot reach the destination. Nonempty path P= $(v_k, v_{k-1}, ...v_{k-m})$ and Q = $(v_{k-m},v_{k-m-1}, ..., v_{k-m}, ..., v_0)$ can be concatenated as follows PQ= $(v_k,v_{k-1}, ..., v_0)$ but concatenation with the empty path returns the path itself: $\epsilon P = P\epsilon = P$.

The set $\mathcal{P}^v$ denotes the permitted paths from v to destination (node v_0), for each $v \in V - \{v_0\}$ and $\mathcal{P}$ be the union of all sets $\mathcal{P}^v$. For each $v \in V - \{v_0\}$, there is a non negative ranking function λ^v, which represent how node v ranks its permitted paths $\mathcal{P}^v$. More preferable paths will have higher values of λ^v. Let $\Lambda = \{ \lambda^v \mid v \in V - \{v_0\}\}$.

An instance of the Stable Path Problem, $S = (G, \mathcal{P}, \Lambda)$, is a graph together with the permitted paths at each node and the ranking function at each node if the following conditions are satisfied for all $v \in V - \{v_0\}$:

1. Empty path is permitted: $\epsilon \in \mathcal{P}^v$
2. Empty path is the lowest ranked path: $\lambda^v(\epsilon) = 0$
3. Strictness: If $\lambda^v(P_1) = \lambda^v(P_2)$ and $P_1 \neq P_2$, then $P_1 = (v, u) P'_1$ and $P_2 = (v, u) P'_2$ for some node u.
4. Simplicity: If path $P \in \mathcal{P}^v$, then P does not have repeated nodes.

Let $S = (G, \mathcal{P}, \Lambda)$ be an instance of the Stable Path Problem. Given a node u, and $W \subseteq \mathcal{P}^u$ such that each path in W has distinct next hops. So the best path in W, best (W, u), is defined to be the highest ranked path in W. A path assignment is a function π that maps each node $u \in V$ to a permitted path $\pi(u) \in \mathcal{P}^u$. Given a path assignment function π and a node u, the set of permitted paths that are one-hop extension of paths through neighbours is defined as

$\text{Choices}(\pi, u) = \{(u, v)\pi(v) \mid \{u, v\} \in E\} \cap \mathcal{P}^u$

So Choices (π, u) represents all possible permitted paths at u that can be formed by extending the paths assigned to the peers of u.

The path assignment π is called stable at node u if

$\pi(u) = \text{best}(\text{choices}(\pi, u), u)$.

The path assignment π is called stable if it is stable at every node $u \in V$.

An Stable Path Problem instance S = (G, P, Λ) is solvable if there exists a stable path assignment π for S. Every such assignment is called a solution for S and written as (P_1, P_2, P_3...,P_n), where π(u) = $P_{u.}$ An instance of SPP may have no solution, or one or more solutions.

3.2.2 Simple Path Vector Protocol (SPVP)

Griffin and Wilfong [21] describe Simple Path Vector Protocol (SPVP) as a distributed algorithm for solving the Stable Path Problem (SPP). SPVP is an abstract version of BGP. Every node runs a copy of the SPVP process. In SPVP, the message exchanged between peers are simply paths. When a node u select a path P $\in \mathcal{P}^u$ as best path it advertise to each w, peers of u.

Each node maintains two data structures:

- rib(u) is the current path that node u is using to reach the destination
- rib_in(u $\Leftarrow$ w) denotes the path that has been most recently advertised by peer w and processed at node u.

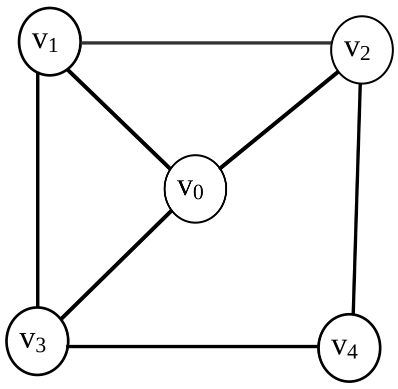
The set of paths available at node u is updated as

Choices(u) = {(u w)P $\in \mathcal{P}^u$ | P = rib_in(u $\Leftarrow$ w)}
and the best path at u is
best(u) = best (choices(u), u).

Neighboring nodes keep exchanging paths that they have currently stored in their rib field. As long as node u receives advertisements from its peers, best(u) is recomputed with the most recent choices(u). The node tries to maintain the most preferred path from the set of available paths in choices(u) as its current path, which is stored in rib(u). Just as it is the case with BGP, when u changes its current path, it notifies its peers about the change. This may cause the peers to send advertisements to their peers, and so on. The network reaches a stable state when there is no node which would change its current path to the destination. If such a static is reached, then the resulting state is the solution of the Stable Paths Problem (SPP). If the Stable Paths Problem (SPP) has no solution, then SPVP diverge.

3.2.3 Example

An example is Good Gadget [21] as shown in fig 3.8. Possible paths and there ranking of each node are shown in the box beside. So the single solution of this Good Gadget is (($v_1v_3v_0$), (v_2v_0), (v_3v_0), ($v_4v_3v_0$)). This is an example SPP problem which has a solution but not a shortest path tree.



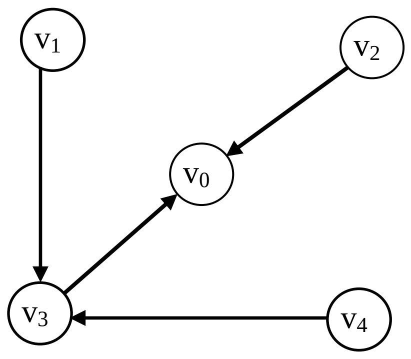
Good Gadget

v_1: $v_1v_3v_0 < v_1v_0$

v_2: $v_2v_1v_0 < v_2v_0$

v_3: v_3v_0

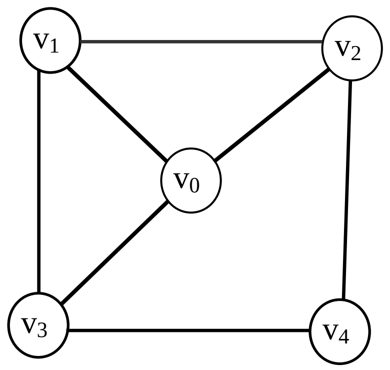
v_4: $v_4v_3v_0 < v_4v_2v_0$



Path Assignment

Figure 3.9: Stable Path Problem with solution - Good Gadget

The example shown in fig 3.9 is a Stable Path Problem (SPP) called Bad Gadget, which has no solution and it causes SPVP to diverge always. The sequence of network states associated with path assignments are shown in figure 3.10.



v_1: $v_1v_3v_0 < v_1v_0$

v_2: $v_2v_1v_0 < v_2v_0$

v_3: $v_3v_4v_2v_0 < v_3v_0$

v_4: $v_4v_2v_0 < v_4v_3v_0$

Bad Gadget

Figure 3.10: Stable Path Problem with no solution - Bad Gadget

Step	π
0	(v_1v_0) (v_2v_0) ($v_3v_4v_2v_0$) ($v_4v_2v_0$)
1	(v_1v_0) ($v_2v_1v_0$) ($v_3v_4v_2v_0$) ($v_4v_2v_0$)
2	(v_1v_0) ($v_2v_1v_0$) ($v_3v_4v_2v_0$) ϵ
3	(v_1v_0) ($v_2v_1v_0$) (v_3v_0) ϵ
4	(v_1v_0) ($v_2v_1v_0$) (v_3v_0) ($v_4v_3v_0$)
5	($v_1v_3v_0$) ($v_2v_1v_0$) (v_3v_0) ($v_4v_3v_0$)
6	($v_1v_3v_0$) (v_2v_0) (v_3v_0) ($v_4v_3v_0$)
7	($v_1v_3v_0$) (v_2v_0) (v_3v_0) ($v_4v_2v_0$)
8	($v_1v_3v_0$) (v_2v_0) ($v_3v_4v_2v_0$) ($v_4v_2v_0$)
9	(v_1v_0) (v_2v_0) ($v_3v_4v_2v_0$) ($v_4v_2v_0$)

Figure 3.11: Sequence of path assignment for Bad Gadget

3.2.4 Solution Using Precedence Metric

Ee et. al. [14] begin by augmenting BGP's decision process, prepending it with an additional step that utilizes a new metric which we call the precedence metric. Each route advertisement is tagged with a global precedence value that is non-negative: a numerically greater value translates to a lower precedence. We denote the precedence value, say x, associated with path P by (P :x). Each AS maintains a history of observed route advertisements from its immediate neighbors. In this history, we associate every route with a local precedence value starting from 0. This local precedence value is obtained from the route's rank, and is determined via the usual BGP decision process. Thus the route ranked i^{th} has a local precedence of i-1 and is preferred over all routes with local precedence greater than that. In short, the selected route is first determined using the incoming global precedence value (since this step occurs before the current BGP decision process), followed by its local precedence value.

Suppose the selected route has an incoming global precedence of t, and a local precedence value of j. Then, the outgoing route advertisement is tagged with a global value of t+j. Thus, a route that is most preferred for all ASes along its path is tagged with 0 at all hops. The destination AS advertises routes with global precedence value of 0.

Following are the main feature that follows the precedence metric approach:

- A history of observed route advertisement
- Precedence Metric consist of global and local values
- First, look at global precedence value to select best route
- If global value are all same, Look at local value instead
- Advertise best route with precedence (global + local)
- More preferred infeasible routes are stored temporarily and incremented the precedence value.
- Maximum period of time required for infeasible route to be stored is N_d * MRAI (Minimum Route Advertisement Interval)
- N_d : number of node around the wheel (rim node included)

Following is a well-known triangular network that experience the route oscillation due to local policy.

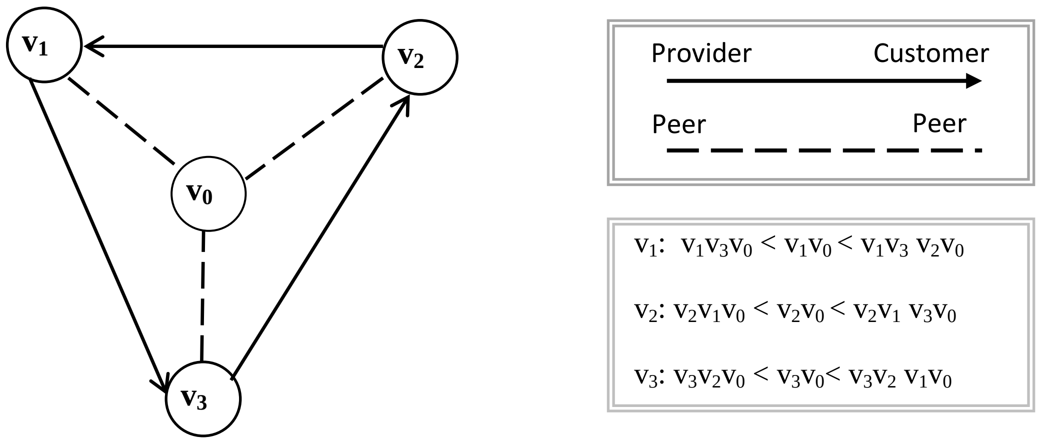

v_1
v_2
v_0
v_3
Provider
Customer
Peer
Peer
v_1: $v_1v_3v_0 < v_1v_0 < v_1v_3\ v_2v_0$
v_2: $v_2v_1v_0 < v_2v_0 < v_2v_1\ v_3v_0$
v_3: $v_3v_2v_0 < v_3v_0 < v_3v_2\ v_1v_0$

Figure 3.12: A triangular network and Local policy

We can describe the solution of the problem in fig. 3.12 as follows:

Step	Node	Best Path	History	Global Precedence	Local Precedence
0	v_1	$v_1\ v_0$	$v_1\ v_0$	0	0
	v_2	$v_2\ v_0$	$v_2\ v_0$	0	0
	v_3	$v_3\ v_0$	$v_3\ v_0$	0	0
1	v_1	$v_1\ v_3\ v_0$	$v_1\ v_3\ v_0$ $v_1\ v_0$	0 0	0 1
	v_2	$v_2\ v_1\ v_0$	$v_2\ v_1\ v_0$ $v_2\ v_0$	0 0	0 1
	v_3	$v_3\ v_2\ v_0$	$v_3\ v_2\ v_0$ $v_3\ v_0$	0 0	0 1
2	v_1	$v_1\ v_0$	$v_1\ v_3\ v_0$ $v_1\ v_0$	0 0	0 1
	v_2	$v_2\ v_0$	$v_2\ v_1\ v_0$ $v_2\ v_0$	0 0	0 1
	v_3	$v_3\ v_0$	$v_3\ v_2\ v_0$ $v_3\ v_0$	0 0	0 1
3	v_1	$v_1\ v_0$	$v_1\ v_3\ v_0$ $v_1\ v_0$	1 0	0 1
	v_2	$v_2\ v_0$	$v_2\ v_1\ v_0$ $v_2\ v_0$	1 0	0 1
	v_3	$v_3\ v_0$	$v_3\ v_2\ v_0$ $v_3\ v_0$	1 0	0 1

Figure 3.13: Step by Step solution of the problem in fig. 3.11 according to the Precedence Metric

The figure 3.12 is a step by step presentation of this solution. In step-0 autonomous systems v_1, v_2 and v_3 all received advertisement from v_0 with global precedence 0 and everyone assign local precedence 0, because only on feasible path in the history now. In step-1 v_1 received advertisement from $v_{3,}$ v_2 received from v_1 and v_3 received from v_2 with global precedence 0, which is the addition of local precedence (0) and global precedence (0) of step 0. Now there are 2 feasible paths so every node assign local precedence 0 and 1 based on path ranking. Same thing happened in step-2. In step-3 all the nodes received advertisement from peers with global precedence 1 (0+1) and settle down in shortest AS path because of lower global precedence but those paths are not the highest ranked path based on local policy. So this solution does not preserve local policy for any of the ASes and also need memory overhead for both history and precedence.

3.2.5 Enforcing Convergence

Cobb and Musunuri [11, 12] proposed a solution for policy based stable path problems. A protocol consists of a set of nodes V and a set of undirected edges E. The node u is a neighbour of node v iff (u,v) ∈ E. In this solution each node do three actions based on different conditions. First, the cost of a node is never smaller than the cost of its next node along its path to root. Next, the best path for a node u has rank lower than it's chosen path π(u), and u will have a next hop along the best path. So π(u) is updated to the best path and because the path of u decreases rank, u increases its cost by one. Finally, it is enabled when the best path of the node is ranked higher than its current path π(u). In this case, node u must also update π(u) to the best path, but it simply sets its cost to the cost of the next node along the new best path. This action also enabled when the rank of the best path of u is lower than the rank of its current path, but in this case the next node is the same in both paths. So the same assignment of values are performed as before. The node chooses to take the new path only if the cost of the next-hop neighbor along the new path is less than a maximum threshold value. If not the node keeps the current path. The main features of the protocol are:

- If the current path of u, i.e., π(u), is different from the best path for u, i.e, best(u, π), the command of the action sets π(u) to best(u, π).
- A new field called cost should be added to the update message.
- Each node is assigned an integer cost. The cost of the node increase whenever its new path has a lower rank than its previous path.
- This new field should be added to the update messages. Therefore, a message in the algorithm is a pair of (P, cost), where p is the path and cost is the cost value.
- As these costs increase a node infer that divergence is occurring.
- A node rejects better path if cost of next-hop node reaches the threshold value.

Let the threshold of the following example is 2, so all nodes stabilize to their lowest preferred paths. So the solution of the oscillation problem in figure 3.11 can be described as figure 3.13 according to this algorithm:

Step	Node	Cost	Best Path
0	v_1	0	$v_1\ v_0$
	v_2	0	$v_2\ v_0$
	v_3	0	$v_3\ v_0$
1	v_1	0	$v_1\ v_3\ v_0$
	v_2	0	$v_2\ v_1\ v_0$
	v_3	0	$v_3\ v_2\ v_0$
2	v_1	1	$v_1\ v_0$
	v_2	1	$v_2\ v_0$
	v_3	1	$v_3\ v_0$
3	v_1	1	$v_1\ v_3\ v_0$
	v_2	1	$v_2\ v_1\ v_0$
	v_3	1	$v_3\ v_2\ v_0$
4	v_1	2	$v_1\ v_0$
	v_2	2	$v_2\ v_0$
	v_3	2	$v_3\ v_0$

Figure 3.14: Step by Step solution of the problem in fig. 3.11 according to the Cobb & Musunuri

Figure 3.13 describe the algorithm proposed by Cobb and Musunuri [11, 12] based on the problem we discussed in figure 3.11. In step-2 cost of the nodes increase from 0 to 1 because the rank of the best path is lower the than the rank of the best path in step-1, according to the algorithm cost of a node will increase only if the new best path has lower rank than previous best path. Again in step-4 cost increased to 2 since the rank of the current best path is lower than the rank of the best path in step-3. In step-4 all the nodes reach to its threshold value 2 and stabilized to their lowest preferred paths. So this solution also can't preserve the local policy and needs memory overhead for both history and cost as well.

3.2.6 Proposed Solution for Inter-Domain Policy Disputes

Consider a network G = (V, E), each node $v_i \in V$ corresponds to an autonomous system, where i={0,1,2,.........,n} and each edge (v_i v_j) ∈ E corresponds to a BGP session between ASes v_i to $v_{j,}$, where i={0,1,2,.........,n} and j={0,1,2,.........,n} but $i \neq j$. We assume that node v_0 is the origin, it is the destination to which all other nodes attempt to establish a path but local preferences of their paths to destination v_0 are different. A path *p* is an edge or a sequence of edges or sequence of connected nodes, (v_i v_j ... ... v_0), i,j > 0 and $i \neq j$. If path p_1 in $\mathcal{P}^v$, set of permitted path is preferable to path p_2 in $\mathcal{P}^v$, we rank as: $r(p_1) < r(p_2)$ where r is a rank function. If the current best path of an AS v, i.e., $\pi(v)$, is different from the highest ranked path in all feasible path of v, i.e, best(v, π), best path is assigned to current highest ranked path. We introduce a parameter Count, that will increase every time best path changes until it reaches to the threshold value. As soon as it reaches to its threshold value the AS will change the rank of AS path or select its best route based on Shortest AS path which is a Well-known mandatory attribute to avoid the oscillation and will advertise this best path as well as threshold value and reset its count. If any AS get an update with threshold value it will reset counter and will work as regular BGP best route selection process. The fundamental characteristics of our proposed algorithm is as follows:

- $\pi(u)$ denotes the path currently chosen as best path by node u.
- best (u, π) represents highest ranked path in all feasible path.
- Count is an integer number and will increase every time the best path change. That means how many times the best path has changed for a particular prefix within an AS. So Count (u) means the number of best path changes of node u.
- T is the threshold value till that count will increase.
- If the count of any router reaches its Threshold value it will change the rank of path based on shortest AS path and will advertise both current best route and count value and reset his count.
- Finally any router receive route with count value which is equal to Threshold, the router will reset the counter and will select the best route based on regular BGP best path selection process.

Figure 3.17 shows our algorithm and we describe the algorithm step by step in figure 3.18. based on the common example in figure 3.11.

```
node u
Begin
if  $\pi$(u) $\neq$ best (u, $\pi$) $\Lambda$ (count (neighbour ) $\neq$ T)  then
        Begin
                $\pi$(u) := best (u, $\pi$)
                count (u) := count (u) + 1
                if count (u) < T  then
                        Advertise ($\pi$(u))
                else
                        Begin
                        $\pi$(u) := Shortest AS path route    // change the rank of path
                        Advertise ($\pi$(u) and count (u))     // based on shortest AS path
                        Reset count (u)
                        End
        End
If  $\pi$(u) $\neq$ best (u, $\pi$) $\Lambda$ (count (neighbour ) = T)  then
        Begin
        $\pi$(u) := best (u, $\pi$)
        reset count (u)
        Advertise ($\pi$(u) and count (neighbour))
        End
End
```

Figure3.15: Our Proposed Algorithm

So the solution of the oscillation problem in figure. 3.11 can be describe as follows according to our proposed algorithm:

Step	**Node**	**BGP Table**	**Best Route**	**Counter**
0	v_1 v_2 v_3	v_1 v_0 v_2 v_0 v_3 v_0	v_1 v_0 v_2 v_0 v_3 v_0	0 0 0
1	v_1	v_1 v_0 v_1 v_3 v_0	v_1 v_3 v_0	1
	v_2	v_2 v_0 v_2 v_1 v_0	v_2 v_1 v_0	1
	v_3	v_3 v_0 v_3 v_2 v_0	v_3 v_2 v_0	1
2	v_1	v_1 v_0 v_1 v_3 v_2 v_0	v_1 v_0	2
	v_2	v_2 v_0 v_2 v_1 v_3 v_0	v_2 v_0	2
	v_3	v_3 v_0 v_3 v_2 v_1 v_0	v_3 v_0	2
3	v_1	v_1 v_0 v_1 v_3 v_0	v_1 v_0 (Shortest AS Path)	3
	v_2	v_2 v_0 v_2 v_1 v_0	v_2 v_1 v_0	0
	v_3	v_3 v_0 v_3 v_2 v_0	v_3 v_0	0

Figure 3.16: Step by Step solution of the problem in figure 3.11 according to our proposed solution

In the figure 3.15 from step-0 to step-2 every time when best path changes count value increases by one. In step-3 if we look at node $v_{1,}$ best path has changed again from v_1 v_0 to v_1 v_3 v_0 so count has increased from 2 to 3. Let the threshold is 3, so count becomes equal to the threshold. Since count and threshold become equal node v_1 changed his best path based on shortest AS path and will advertise this best path and count value or threshold value to its peer. As soon as the peers receive the count value is equal to the threshold it will reset his own count and select the best path based on regular BGP best path selection process. So any one ASes of the disputed topology settle down or break the oscillation other participating ASes must goes to a stable state.

Analysis:

- It does not need to maintain local history consequently it does not need to carry history of path changes in Update Messages.
- It preserve local policy for
  - n/2 ASes if the number of ASes participated in this dispute topology are even number.
  - (n-1)/2 ASes if the number of ASes participated in this dispute topology are odd number.

These are the two main performance metric that we focus to improve in our algorithm. So we can solve the disputed topology in figure 3.11 by applying our proposed solution. If we start from AS v_1 the solved topology will be as figure 3.18. Accordingly we can show that starting from any AS's:

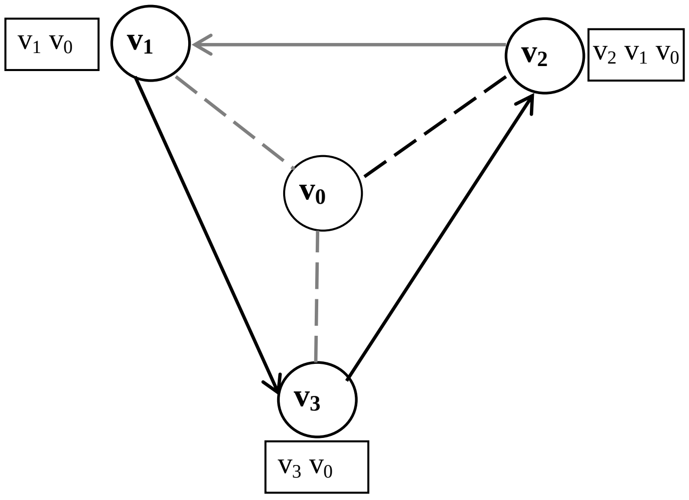


Figure 3.17: Solution of the Oscillation If we start from V_1

3.3: Implementation of Proposed Solution for Inter-Domain Policy Disputes

The proposed algorithm for Inter-Domain policy disputes has successfully implemented by using SimBGP simulator, a lightweight event-driven BGP simulator implemented with Python [35]. Three test case topologies have been created using simBGP Configuration syntax to test algorithm. Among these test cases, test case one comes with simBGP simulator. Test cases with topology configuration and output has shown below case by case.

3.3.1 Test Case One

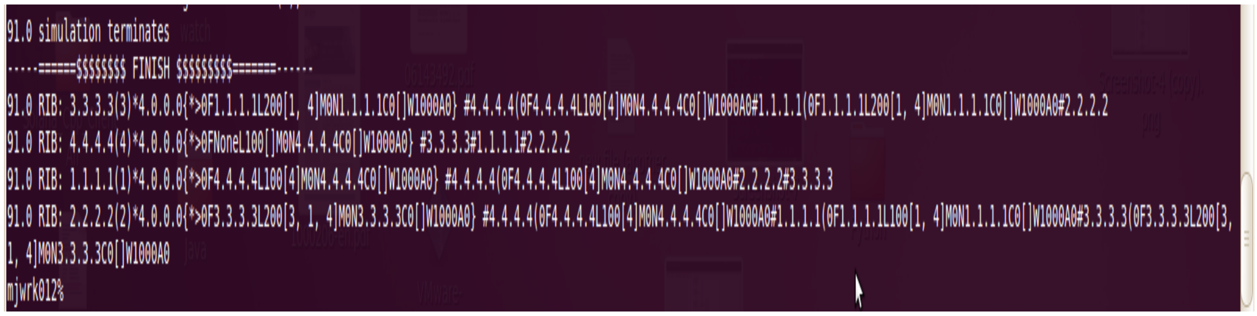
We tested this topology in original simBGP simulator and simulator after implementation of our proposed algorithm. The simBGP original simulator can solve this topology only with this local policy. In this topology with defined local policy after exchanging some number of updates among the ASes, any of the ASes will get its advertisement back. When any of the ASes has received its advertisement back, the original simBGP stop doing any more event and terminate the simulation. After implementation of the algorithm in SimBGP simulator we run the above topology with same local policy have got the desired output without route oscillation after it has reached the threshold. A sample output of our algorithm has shown in figure 3.18 below.

```
!       [1]
!      / | \
!     /  |  \
!    /  [4]  \
!   /  /   \  \
!  [2]--------[3]
!

router bgp 1
 bgp router-id 1.1.1.1
 neighbor 2.2.2.2 remote-as 2
 neighbor 2.2.2.2 advertisement-interval 30
 neighbor 2.2.2.2 route-map 1pref2_2 in
 neighbor 3.3.3.3 remote-as 3
 neighbor 3.3.3.3 advertisement-interval 30
 neighbor 2.2.2.2 route-map 1pref2_3 in
 neighbor 4.4.4.4 remote-as 4
 neighbor 4.4.4.4 advertisement-interval 30
 neighbor 4.4.4.4 route-map 1pref2_4 in
```

```
route-map 1pref2_2 permit 10
 set local-preference 200

route-map 1pref2_3 permit 10
 set local-preference 100

route-map 1pref2_4 permit 10
 set local-preference 100

router bgp 2
 bgp router-id 2.2.2.2
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 advertisement-interval 30
 neighbor 1.1.1.1 route-map 2pref3_1 in
 neighbor 3.3.3.3 remote-as 3
 neighbor 3.3.3.3 advertisement-interval 30
 neighbor 3.3.3.3 route-map 2pref3_3 in
 neighbor 4.4.4.4 remote-as 4
 neighbor 4.4.4.4 advertisement-interval 30
 neighbor 4.4.4.4 route-map 2pref3_4 in

route-map 2pref3_1 permit 10
 set local-preference 100

route-map 2pref3_3 permit 10
 set local-preference 200

route-map 2pref3_4 permit 10
 set local-preference 100

router bgp 3
 bgp router-id 3.3.3.3
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 advertisement-interval 30
 neighbor 1.1.1.1 route-map 3pref1_1 in
 neighbor 2.2.2.2 remote-as 2
 neighbor 2.2.2.2 advertisement-interval 30
 neighbor 2.2.2.2 route-map 3pref1_2 in
 neighbor 4.4.4.4 remote-as 4
 neighbor 4.4.4.4 advertisement-interval 30
 neighbor 4.4.4.4 route-map 3pref1_4 in

route-map 3pref1_1 permit 10
 set local-preference 200

route-map 3pref1_2 permit 10
 set local-preference 100

route-map 3pref1_4 permit 10
 set local-preference 100

router bgp 4
 bgp router-id 4.4.4.4
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 advertisement-interval 30
 neighbor 2.2.2.2 remote-as 2
 neighbor 2.2.2.2 advertisement-interval 30
 neighbor 3.3.3.3 remote-as 3
 neighbor 3.3.3.3 advertisement-interval 30

event announce-prefix 4.4.4.4 4.0.0.0  2.0
event terminate 100

debug show-receive-events
debug show-final-ribs
```

Figure 3.18: Test Case One Topology and Configuration

Output:



```
91.0 simulation terminates
-----======$$$$$$$$ FINISH $$$$$$$$$=======------
91.0 RIB: 3.3.3.3(3)*4.0.0.0{*>0F1.1.1.1L200[1, 4]M0N1.1.1.1C0[]W1000A0} #4.4.4.4(0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0#1.1.1.1(0F1.1.1.1L200[1, 4]M0N1.1.1.1C0[]W1000A0#2.2.2.2
91.0 RIB: 4.4.4.4(4)*4.0.0.0{*>0FNoneL100[]M0N4.4.4.4C0[]W1000A0} #3.3.3.3#1.1.1.1#2.2.2.2
91.0 RIB: 1.1.1.1(1)*4.0.0.0{*>0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0} #4.4.4.4(0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0#2.2.2.2#3.3.3.3
91.0 RIB: 2.2.2.2(2)*4.0.0.0{*>0F3.3.3.3L200[3, 1, 4]M0N3.3.3.3C0[]W1000A0} #4.4.4.4(0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0#1.1.1.1(0F1.1.1.1L100[1, 4]M0N1.1.1.1C0[]W1000A0#3.3.3.3(0F3.3.3.3L200[3,
1, 4]M0N3.3.3.3C0[]W1000A0
mjwrk012%
```

Figure 3.19: Output of Test Case One

According to the output in figure 3.19 node v_1 (1.1.1.1) select $\{v_1v_4\}$ as its best path. node v_3 (3.3.3.3) select $\{v_3v_1v_4\}$ as its best path because of its highest local preference (200) among all feasible routes. Finally node v_2 (2.2.2.2) takes $\{v_2v_3v_1v_4\}$ as its best path because the local preference of this path is 200, which is the highest.

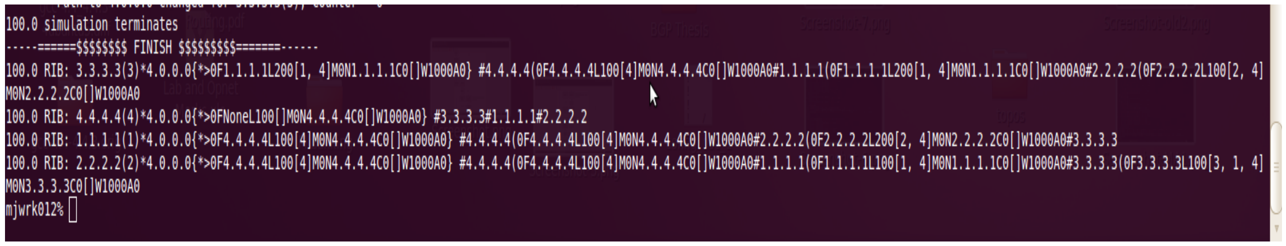
3.3.2 Test Case Two

This test case has developed maintaining the same characteristics as we describe in section 3.1.1 and the common topology for oscillation shown in figure 3.11. This topology has also tested in both original and new simulators implementing our algorithm. The old simulator cannot resolve the oscillation of this topology but the new simulator after implementing our algorithm can successfully solve the oscillation problem. The topology with configuration and output has shown below.

```
!       [1]
!      / | \
!     /  |  \
!    /  [4]  \
!   /  /   \  \
!  [2]--------[3]
!

router bgp 1
 bgp router-id 1.1.1.1
 neighbor 2.2.2.2 remote-as 2
 neighbor 2.2.2.2 advertisement-interval 30
 neighbor 2.2.2.2 route-map 1pref2_2 in
 neighbor 3.3.3.3 remote-as 3
 neighbor 3.3.3.3 advertisement-interval 30
 neighbor 4.4.4.4 remote-as 4
 neighbor 4.4.4.4 advertisement-interval 30

route-map 1pref2_2 permit 10
 match as-path 2_4
 set local-preference 200

router bgp 2
 bgp router-id 2.2.2.2
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 advertisement-interval 30
 neighbor 3.3.3.3 remote-as 3
 neighbor 3.3.3.3 advertisement-interval 30
 neighbor 3.3.3.3 route-map 2pref3_3 in
 neighbor 4.4.4.4 remote-as 4
 neighbor 4.4.4.4 advertisement-interval 30

route-map 2pref3_3 permit 10
 match as-path 3_4
 set local-preference 200

router bgp 3
 bgp router-id 3.3.3.3
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 advertisement-interval 30
 neighbor 1.1.1.1 route-map 3pref1_1 in
 neighbor 2.2.2.2 remote-as 2
 neighbor 2.2.2.2 advertisement-interval 30
 neighbor 4.4.4.4 remote-as 4
 neighbor 4.4.4.4 advertisement-interval 30

route-map 3pref1_1 permit 10
 match as-path 1_4
 set local-preference 200

router bgp 4
 bgp router-id 4.4.4.4
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 advertisement-interval 30
 neighbor 2.2.2.2 remote-as 2
 neighbor 2.2.2.2 advertisement-interval 30
 neighbor 3.3.3.3 remote-as 3
 neighbor 3.3.3.3 advertisement-interval 30

event announce-prefix 4.4.4.4 4.0.0.0  2.0
event terminate 100

debug show-receive-events
debug show-final-ribs
```

Figure 3.20: Test Case Two Topology and Configuration

Output:



```
100.0 simulation terminates
-----======$$$$$$$$ FINISH $$$$$$$$$=======------
100.0 RIB: 3.3.3.3(3)*4.0.0.0{*>0F1.1.1.1L200[1, 4]M0N1.1.1.1C0[]W1000A0} #4.4.4.4(0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0#1.1.1.1(0F1.1.1.1L200[1, 4]M0N1.1.1.1C0[]W1000A0#2.2.2.2(0F2.2.2.2L100[2, 4]
M0N2.2.2.2C0[]W1000A0
100.0 RIB: 4.4.4.4(4)*4.0.0.0{*>0FNoneL100[]M0N4.4.4.4C0[]W1000A0} #3.3.3.3#1.1.1.1#2.2.2.2
100.0 RIB: 1.1.1.1(1)*4.0.0.0{*>0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0} #4.4.4.4(0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0#2.2.2.2(0F2.2.2.2L200[2, 4]M0N2.2.2.2C0[]W1000A0#3.3.3.3
100.0 RIB: 2.2.2.2(2)*4.0.0.0{*>0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0} #4.4.4.4(0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0#1.1.1.1(0F1.1.1.1L100[1, 4]M0N1.1.1.1C0[]W1000A0#3.3.3.3(0F3.3.3.3L100[3, 1, 4]
M0N3.3.3.3C0[]W1000A0
mjwrk012% 
```

Figure 3.21: Output of Test Case Two

The output of test case two in figure 3.21 shows that node v_1 (1.1.1.1) select $\{v_1v_4\}$ as its best path though the path $v_1v_2v_4$ has the highest local preference (200), because this AS detect the oscillation first and changes the rank of the paths based on shortest AS path. node v_3 (3.3.3.3) select $\{v_3v_1v_4\}$ as its best path because of its highest local preference (200) among all feasible routes so this node preserve its local policy. Finally node v_2 (2.2.2.2) takes $\{v_2v_4\}$ as its best path because all the feasible routes has same local preference (100).

3.3.3 Test Case Three

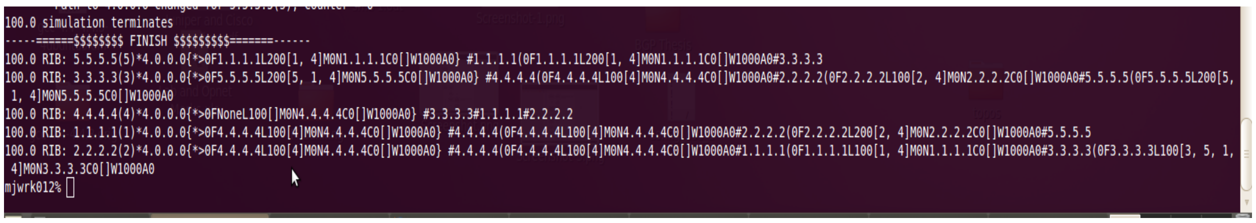
In this topology we added one more AS, AS5 and this AS does not connected to the AS4, the source AS. The local policy has assigned maintaining the characteristics as we describe in section 3.1.1. We also run this test case to both original and new simulator. The result is as same as test case 2, the original simulator cannot solve this problem but the simulator based on our algorithm successfully solve the oscillation. The topology with configuration has shown in figure 3.21 and out has shown in figure 3.22 .

```
!         [1]
!        / | \
!       /  |   \
!      /   |     \
!     /   [4]     [5]
!    /  /   \     /
!   / /       \  /
!  [2]--------[3]
!

router bgp 1
 bgp router-id 1.1.1.1
 neighbor 2.2.2.2 remote-as 2
 neighbor 2.2.2.2 advertisement-interval 30
 neighbor 2.2.2.2 route-map 1pref2_2 in
 neighbor 5.5.5.5 remote-as 5
 neighbor 5.5.5.5 advertisement-interval 30
 neighbor 4.4.4.4 remote-as 4
 neighbor 4.4.4.4 advertisement-interval 30

route-map 1pref2_2 permit 10
 match as-path 2_4
 set local-preference 200

router bgp 2
 bgp router-id 2.2.2.2
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 advertisement-interval 30
 neighbor 3.3.3.3 remote-as 3
 neighbor 3.3.3.3 advertisement-interval 30
 neighbor 3.3.3.3 route-map 2pref3_3 in
 neighbor 4.4.4.4 remote-as 4
 neighbor 4.4.4.4 advertisement-interval 30

route-map 2pref3_3 permit 10
 match as-path 3_4
 set local-preference 200
```

```
router bgp 3
 bgp router-id 3.3.3.3
 neighbor 2.2.2.2 remote-as 2
 neighbor 2.2.2.2 advertisement-interval 30
 neighbor 5.5.5.5 remote-as 5
 neighbor 5.5.5.5 advertisement-interval 30
 neighbor 5.5.5.5 route-map 3pref5_5 in
 neighbor 4.4.4.4 remote-as 4
 neighbor 4.4.4.4 advertisement-interval 30

route-map 3pref5_5 permit 10
 match as-path 5_1_4
 set local-preference 200

router bgp 5
 bgp router-id 5.5.5.5
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 advertisement-interval 30
 neighbor 1.1.1.1 route-map 5pref1_1 in
 neighbor 3.3.3.3 remote-as 3
 neighbor 3.3.3.3 advertisement-interval 30

route-map 5pref1_1 permit 10
 match as-path 1_4
 set local-preference 200

router bgp 4
 bgp router-id 4.4.4.4
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 advertisement-interval 30
 neighbor 2.2.2.2 remote-as 2
 neighbor 2.2.2.2 advertisement-interval 30
 neighbor 3.3.3.3 remote-as 3
 neighbor 3.3.3.3 advertisement-interval 30

event announce-prefix 4.4.4.4 4.0.0.0  2.0
event terminate 100

debug show-receive-events
debug show-final-ribs
```

Figure 3.22: Test Case Three Topology and Configuration

Output:



```
100.0 simulation terminates
-----======$$$$$$$$ FINISH $$$$$$$$$=======------
100.0 RIB: 5.5.5.5(5)*4.0.0.0{*>0F1.1.1.1L200[1, 4]M0N1.1.1.1C0[]W1000A0} #1.1.1.1(0F1.1.1.1L200[1, 4]M0N1.1.1.1C0[]W1000A0#3.3.3.3
100.0 RIB: 3.3.3.3(3)*4.0.0.0{*>0F5.5.5.5L200[5, 1, 4]M0N5.5.5.5C0[]W1000A0} #4.4.4.4(0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0#2.2.2.2(0F2.2.2.2L100[2, 4]M0N2.2.2.2C0[]W1000A0#5.5.5.5(0F5.5.5.5L200[5,
 1, 4]M0N5.5.5.5C0[]W1000A0
100.0 RIB: 4.4.4.4(4)*4.0.0.0{*>0FNoneL100[]M0N4.4.4.4C0[]W1000A0} #3.3.3.3#1.1.1.1#2.2.2.2
100.0 RIB: 1.1.1.1(1)*4.0.0.0{*>0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0} #4.4.4.4(0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0#2.2.2.2(0F2.2.2.2L200[2, 4]M0N2.2.2.2C0[]W1000A0#5.5.5.5
100.0 RIB: 2.2.2.2(2)*4.0.0.0{*>0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0} #4.4.4.4(0F4.4.4.4L100[4]M0N4.4.4.4C0[]W1000A0#1.1.1.1(0F1.1.1.1L100[1, 4]M0N1.1.1.1C0[]W1000A0#3.3.3.3(0F3.3.3.3L100[3, 5, 1,
 4]M0N3.3.3.3C0[]W1000A0
mjwrk012% 
```

Figure 3.23: Output of Test Case Three

The output of test case three in figure 3.23 describes that node v_1 (1.1.1.1) selects $\{v_1v_4\}$ as its best path though the path $v_1v_2v_4$ has the highest local preference (200), because this AS detect the oscillation first and changes the rank of the paths based on shortest AS path. Node v_5 (5.5.5.5) selects $\{v_5v_1v_4\}$ as its best path because of highest local preference (200) and preserve its local policy. Node v_3 (3.3.3.3) selects $\{v_3v_5v_1v_4\}$ as its best path because of its highest local preference (200) among all feasible routes so this node also preserve its local policy. Finally node v_2 (2.2.2.2) takes $\{v_2v_4\}$ as its best path because all the feasible routes has same local preference (100).

3.4 Comparison between original simulator and New simulator

Test Cases	Time to Stop Oscillation in Seconds (Original Simulator)	Time to Stop Oscillation in Seconds (New Simulator)
Test Case 1	29.69	28.25
Test Case 2	Infinite	55.07
Test Case 3	Infinite	55.07

Figure 3.24: Comparison between our algorithm and original simBGP

The Original simBGP is a general BGP simulator and they add a special feature to solve the oscillation only the case like test case one [35]. They solve this problem only the case when any AS received his advertisement back as a loop. After the implementation of our algorithm in simBGP it can solve the oscillation due to local policy in any dispute wheel topology. Three different test cases have been tested using our new simulator and we get the expected result. In table 3.23 a comparative study has show between the new simulator and old simulator with respect to three different test cases. Original simulator can solve only the test case one however New simulator can solve all the three test cases. The required time (in second) to stop the oscillation has also mentioned to make it more clear.

Chapter Four

Instability Detection Tool, Test Cases and Results

This chapter describes the tool we proposed and developed to detect route instability. The internet today has more than 400,000 routes distributed among 60,000 ASes and network operators can only monitor BGP decisions in their ASes. So it is very important for the network admin to detect route instability or oscillation is happening there. BGP data has been collected through an iterative process from Route View, our test Network and Simulator using Solana ND. The algorithm is tested for both real BGP data and data from simulated topology and we successfully identify the route oscillation within the AS.

The following figure 4.1 is the step-by-step process how we progress to accomplish this study.

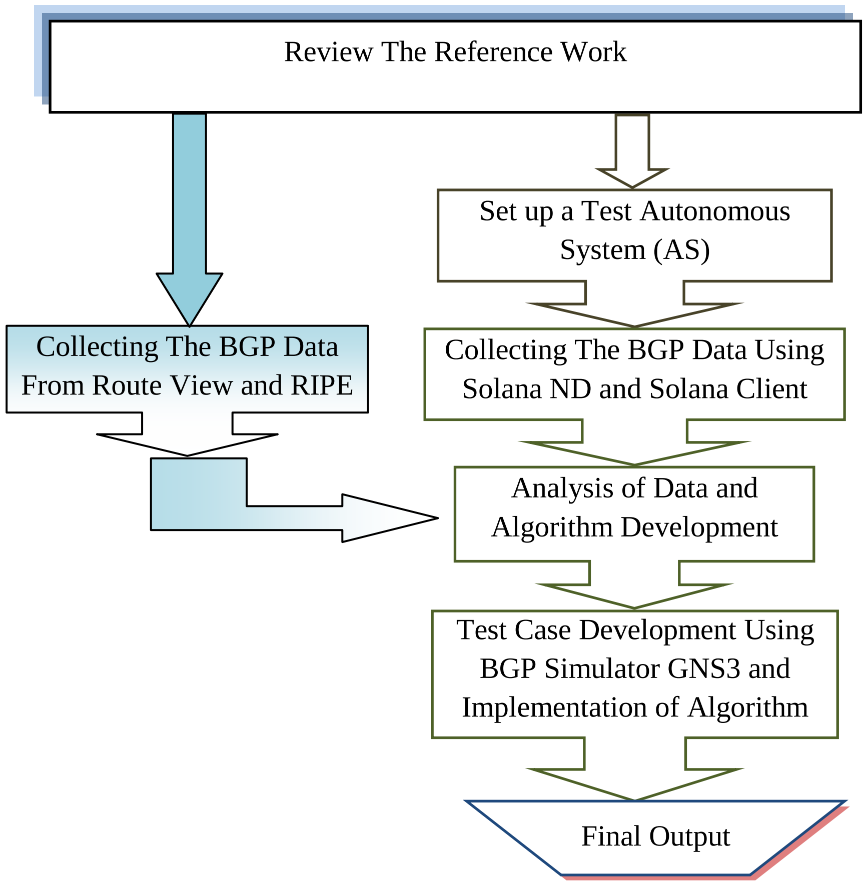

Review The Reference Work
Set up a Test Autonomous System (AS)
Collecting The BGP Data From Route View and RIPE
Collecting The BGP Data Using Solana ND and Solana Client
Analysis of Data and Algorithm Development
Test Case Development Using BGP Simulator GNS3 and Implementation of Algorithm
Final Output

Figure 4.1: Overview of The Research Study

4.1 Routing Data Analysis from public source

Two main public sources of BGP table and update data are Oregon Route Views [31] and RIPE NCC [29]. Both archive daily table dumps as well as NTP-synchronized updates in MRT format, a well-known binary format for archiving BGP data. Such data are often collected by Cisco and the zebra software router, a public GNU based software with several routing functions including OSPF, RIP, and BGP. Zebra conforms to the most recent BGP RFC1771 and supports several BGP extensions. It has been tested to work with several different router vendors with different protocol features and uses the MRT format to store the data. We collect and use Oregon Route View data in our study from route view archives as a public source of BGP routes.

The Route Views Project [30] peers with around 30 networks mostly within the U.S., but some in Europe. Among them there are many tier-1 ISPs such as Level3, Cable&Wireless, AT&T, Sprint, and Verio. RIPE peers mostly come from Europe. A majority of these feeds are configured to be default-free; thus, one can observe the entire routing table from those view points. The motivation behind the tier-1 ISPs to provide such data feeds is to provide easy data access for debugging routing problems. These peering sessions are passive and typically one-directional. The routers from RIPE and Route Views (*monitor router*) only receive the updates from routers in different networks (*operator routers*), but do not inject any routes in the routing session. However, to be defensive against routing misconfigurations, the monitor should configure the export policy so that no routes are allowed to be sent out. Similarly, the operator routers should be configured the import policy for the peering session associated with the monitor routers, with the intention that no routes are accepted. The peers simply provide a view of their routing table and do not provide any connectivity; thus, RIPE or Route Views routers cannot use them to forward traffic. These feeds contain the best routes of the particular routers who peer with Route Views and RIPE.

4.2 Test Network Setup

Although we collect and analyze the Oregon Route View BGP data we also set up a private Autonomous System or a test network tested and connect that to Ryerson University Computing and Communications Services (CCS), a Ryerson university ISP and the AS number is 26996. We use both Cisco and Juniper routers to establish the network which helps us to understand details of the vendor implementations of an Autonomous System. Solana Smart Hawk Network device (ND) is used in to the test network to collect and monitor the BGP routes. Solana Networks FalconView and SmartHawk appliances are intelligent network monitoring and diagnostics platforms that provide network administrators with visibility into Routing within their IP networks. We also use Web Service (WS) Client to collect BGP data as XML file through Solana ND. The WS Client has run on Red Hat Linux Machine and connect that to the Solana ND to collect and save the XML file. Finally we run our software based on instability detection algorithm in Windows environment and generate report.

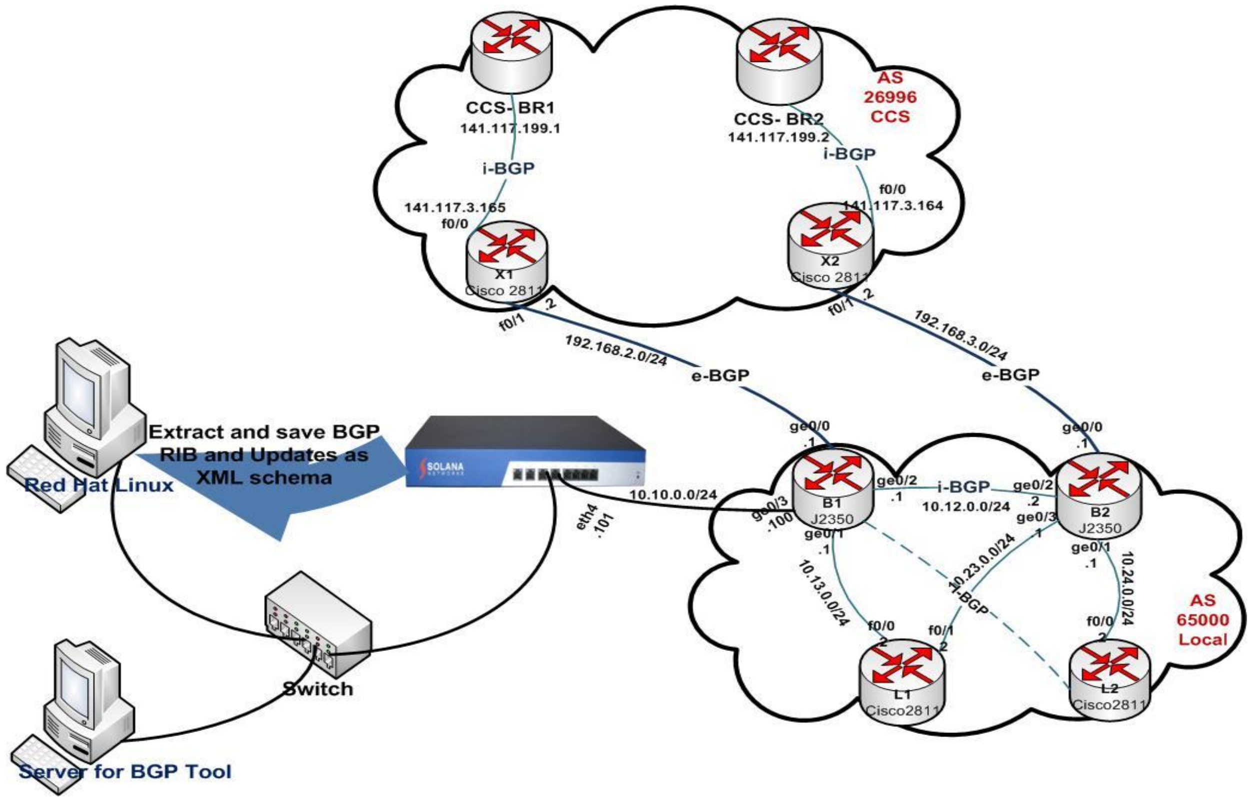


Figure 4.2: Test Network

4.3 BGP Simulator

Measurements through the test Network allow us to study the current routing behavior. Simulation is very helpful to analyze the impact of particular parameter settings, implementation variants of protocols, or the effect of topologies. We simulate BGP protocol in some test topologies to study the effect of route instability due to local policy. Such topologies typically have multiple alternate paths of different AS path lengths, and define their performance based on local policy. We make use of a publicly available BGP simulator GNS3. This simulation package use the real IOS of routers, we use Cisco 7200 series IOS and capable of modeling large, complex networks. We implemented four test cases to study the effect of route instability because of Local Policies of ASes. We provide several parameter settings used as default settings by major router vendors and work on weight, Local Preference and community attributes focusing on route instability and route non-determinism of AS path. Figure 4.4 is a general network topology designed using GNS3.

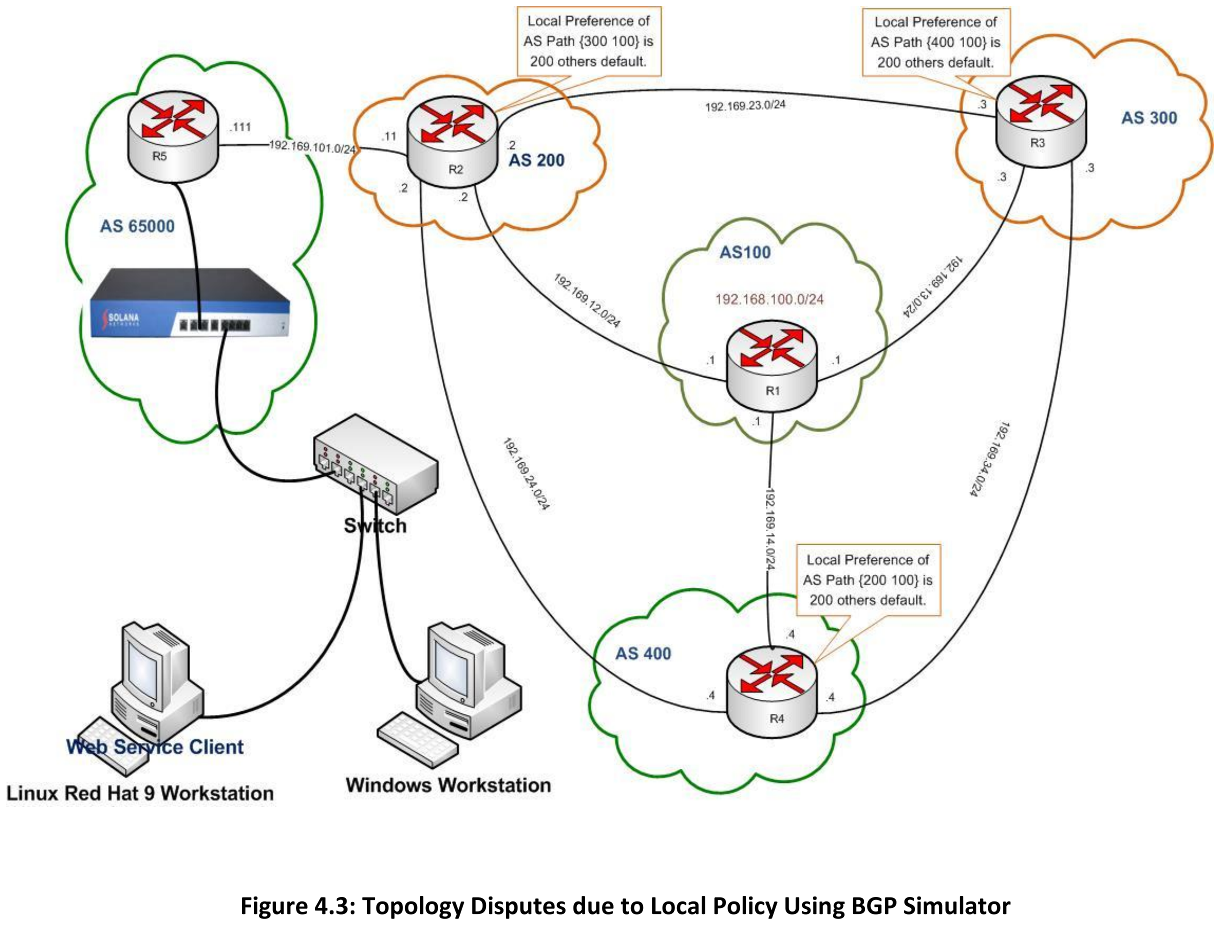


Figure 4.3: Topology Disputes due to Local Policy Using BGP Simulator

4.4 Proposed Fault Detection Algorithm

In this algorithm, we concentrate on BGP local policy induced route instability such as oscillation and non-determinism and to report that to network administrator so that he/she can respond quickly. This will increase customer satisfaction. In this tool, we track the routing changes and transform BGP updates close in time (T_w) for certain prefix into events, the time is a function of the Minimum Route Advertisement Interval (MRAI=30s). The BGP local policy and misconfiguration could create global conflicts or global topology disputes which results route oscillations.

Following are some types of disruptions and their symptoms:

Route oscillation: results in many advertisements (updates and withdraws) within small period of time and changes the best route. In this case, the AS-path and or the Next-hop address of that prefix could change.

Link down: results in withdrawal of all prefixes which are using that link.

Peer down: results in withdrawal of all prefixes that are advertized by that peer. Hence, the AS-path and or the next-hop address could change.

The system is connected to one border router in the local AS and saves the used (best) route for a certain prefix (p) from the routing table of that boarder router. It starts collecting all BGP updates received by this router from all neighboring BGP routers. We will use Solana Network Device (ND) to collect the RIB and updates.

We define a time window (T_w=15 minutes) and create a list for each BGP neighbor in which we save all update messages received from each neighbor by the boarder BGP router within T_w time. The reason behind the selecting T_w min is Solana ND collects update every 15 minutes and save as XML file.

For each prefix (p), we extract the origin, Local Preference, AS path and AS path length, MED value, next hop address, number of withdrawals and number of updates. We count the number of updates received for each prefix from each peer within T_w. If any of these numbers is greater than one, the algorithm will check if the new routes have the same next hop address. If next hop is the same, the algorithm will declare that there is unstable session; otherwise it will declare a probability of oscillation.

If only one update is received within T_w, the algorithm will declare the path has changed.

The Algorithm can be summarized as in the following diagram.

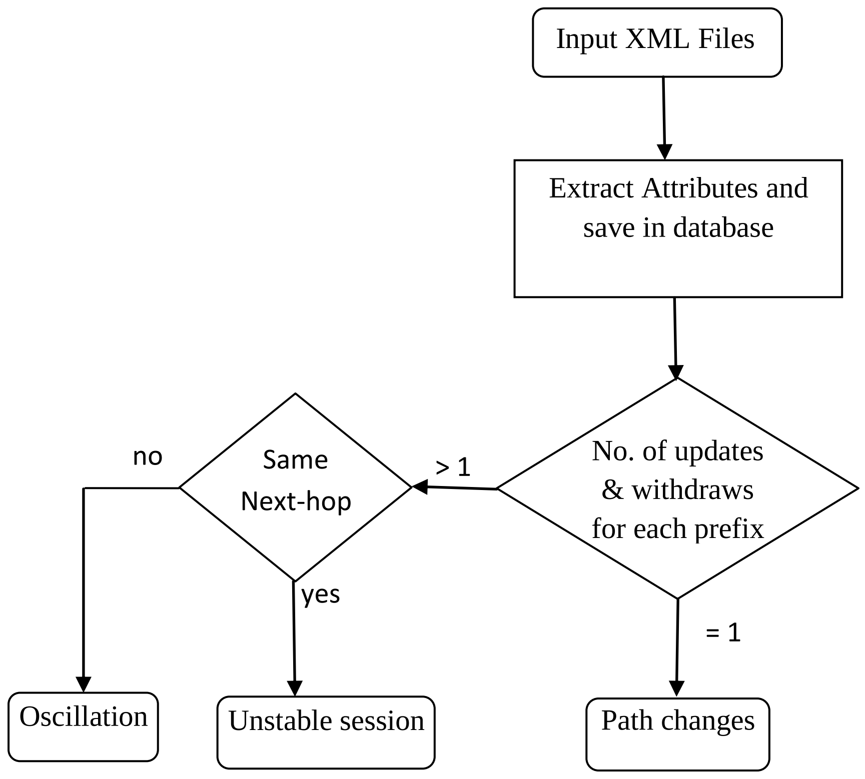

Input XML Files
Extract Attributes and save in database
No. of updates & withdraws for each prefix
> 1
= 1
Same Next-hop
no
yes
Oscillation
Unstable session
Path changes

Figure 4.4: Route Instability Detection Algorithm

4.5 Architecture of the Software

We first collect the BGP RIB and updates as XML file using Solana Network Device. The algorithm has realized through the Java programming language which provided the necessary API's to complete the project. MySQL was selected as the database program for its simplicity and accepted standards. The MySQL database has used due to the large data sets created from parsing the collected XML file. It is recommended to run Program and MySQL server on a single machine as the communication process between both programs is intensive.

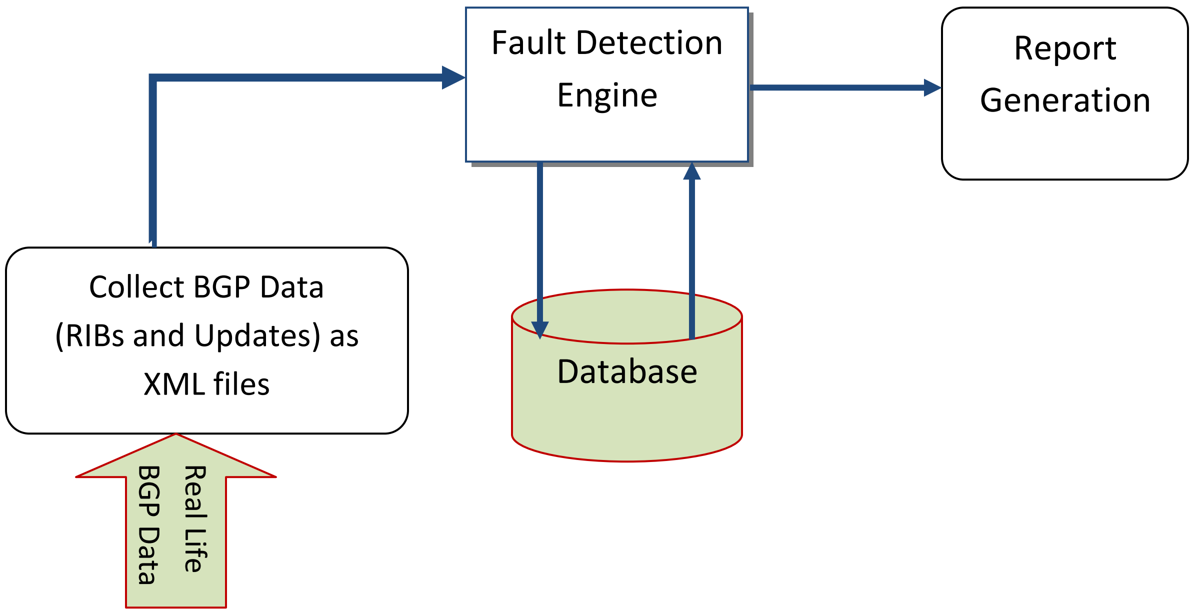


Figure 4.5: Architecture of the route instability detection tool within an AS

4.6 Software Development

We use Web Service Client Software developed by Solana Networks to collect real life BGP routes and updates as an XML file from our test network as well as test cases. After that we use our software or tool developed by using Java to pull out BGP data from XML schema. BGP data are stored in database through MySQL. Finally we implement our algorithm and apply that to these data to find out desired outputs.

The program is separated into five unique classes as seen in the figure below.

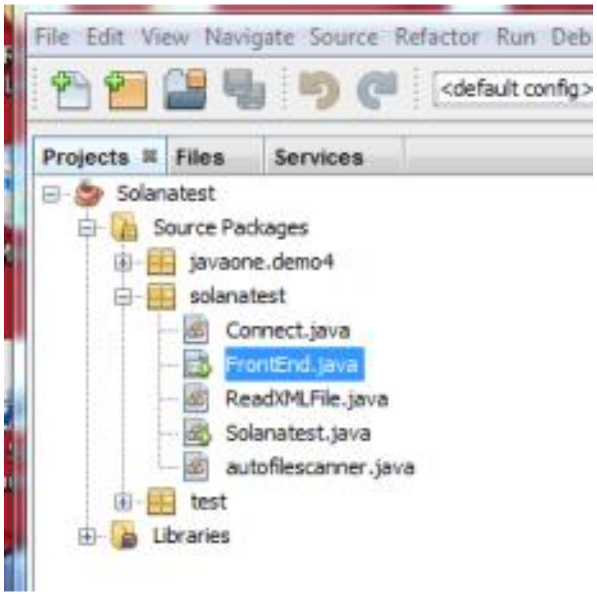


Figure 4.6: Program Structure of route instability detection algorithm

Connect.java is used to create the interface with the MySQL server. In this class there are methods used to define the connection process.

FrontEnd.java class is used to provide the GUI interface for the end user. It includes features such as login and password for the MySQL server and file locations for the xml files generated from the Smart Hawk. The FrontEnd.java is also used to spawn process (autofilescanner.java). The main class is contained in FrontEnd.java.

ReadXMLFile.java is used to parse the xml files generated from the Smart Hawk box.

autofilescanner.java class contains the complete implementation of route oscillation algorithm.

4.7: Test Cases to check The Policy Induced Topology Disputes

4.7.1: Test Case One:

In this topology we construct four ASes, AS 100, AS 200, AS 300 and AS 400 using GNS3 simulator. Another AS 65000 has established using physical routers. Cisco 2811 router and Solana ND has used to construct this AS, This AS is established mainly to collect the BGP RIB and updates those we use as input of our Program. In this test case all the ingress and egress points are in the same router. The weight attribute, is significant into the router locally (Cisco proprietary attribute and considered as first attribute for BGP best route selection process) has used to set the local policy in this test case.

So the local policy for ASes in this topology are :

AS 200: Rank of path {200 300 100} < {200 100} (We set the weight for AS path {300 100} as 200 and keep others default.)

AS 300: Rank of path {300 400 100} < {300 100} (We set the weight for AS path {400 100} as 200 and keep others default.)

AS 400: Rank of path {400 200 100} < {400 100} (We set the weight for AS path {200   100} as 200 and keep others default.)

The lower rank means higher preference.

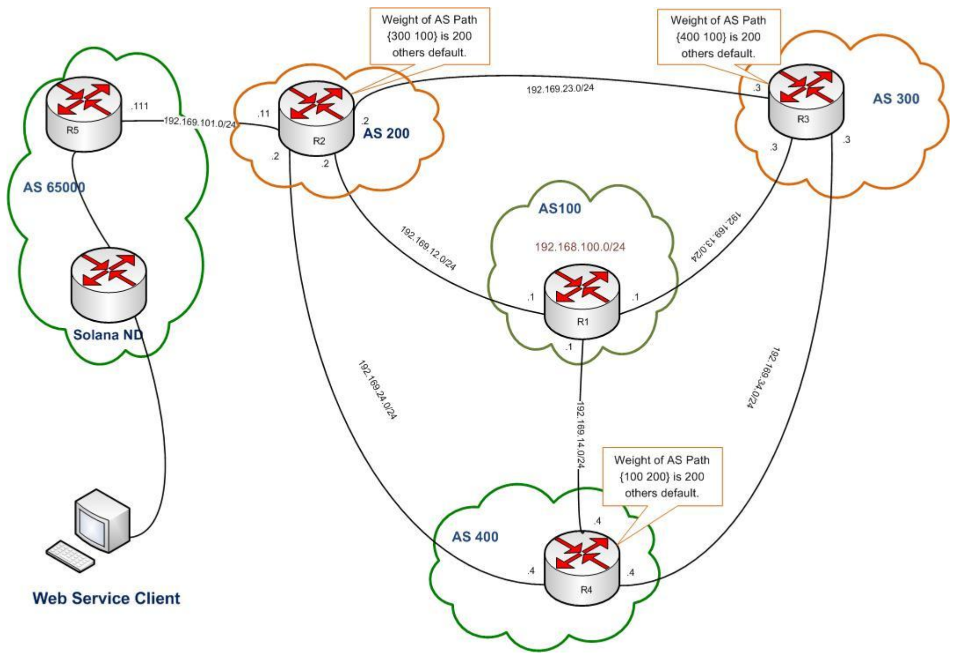


Figure 4.7: Topology for Test Case One manipulating weight attribute

After the completion of all the configuration according to the topology as above we start collecting BGP data and test them using the tool. BGP main table and updates of this topology collected according to our step by step test methodology describe in appendix A has shown below.

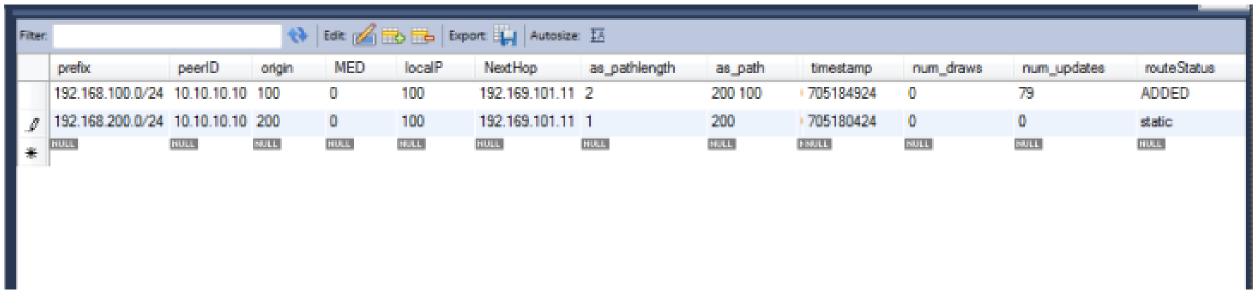
```
<asNumber="65000">
 <Peer>
  <peerID>10.10.10.10</peerID>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>200 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
   <Route>
    <prefix>192.168.200.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>200 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
 </Peer>
</ns:autonomousSystem>
```

Figure 4.8: BGP RIB for Test Case One

```
<asNumber="65000">
 <Peer>
  <peerID>10.10.10.10</peerID>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>200 300 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>200 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
    <pathSegment>
     <path>200 300 100 </path>
    </pathSegment>
   </aspath>
   <nexthop>
    <nextHop>192.169.101.11</nextHop>
   </nexthop>
   <multiexit>
    <multiExit>0</multiExit>
   </multiexit>
   <localpref>
    <localPref>100</localPref>
   </localpref>
   <routeStatus>ADDED</routeStatus>
  </Route>
  <Route>
   <prefix>192.168.100.0/24</prefix>
   <origin>
    <originId>IGP</originId>
   </origin>
   <aspath>
    <pathSegment>
     <path>200 100 </path>
    </pathSegment>
   </aspath>
   <nexthop>
    <nextHop>192.169.101.11</nextHop>
   </nexthop>
   <multiexit>
    <multiExit>0</multiExit>
   </multiexit>
   <localpref>
    <localPref>100</localPref>
   </localpref>
   <routeStatus>ADDED</routeStatus>
  </Route>
  <Route>
   <prefix>192.168.100.0/24</prefix>
   <origin>
    <originId>IGP</originId>
   </origin>
   <aspath>
    <pathSegment>
     <path>200 300 100 </path>
    </pathSegment>
   </aspath>
   <nexthop>
    <nextHop>192.169.101.11</nextHop>
   </nexthop>
   <multiexit>
    <multiExit>0</multiExit>
   </multiexit>
   <localpref>
    <localPref>100</localPref>
   </localpref>
   <routeStatus>ADDED</routeStatus>
  </Route>
  <Route>
   <prefix>192.168.100.0/24</prefix>
   <origin>
    <originId>IGP</originId>
   </origin>
   <aspath>
    <pathSegment>
     <path>200 100 </path>
    </pathSegment>
   </aspath>
   <nexthop>
    <nextHop>192.169.101.11</nextHop>
   </nexthop>
   <multiexit>
    <multiExit>0</multiExit>
   </multiexit>
   <localpref>
    <localPref>100</localPref>
   </localpref>
   <routeStatus>ADDED</routeStatus>
  </Route>
```

Figure 4.9: BGP Sample Update for Test Case One

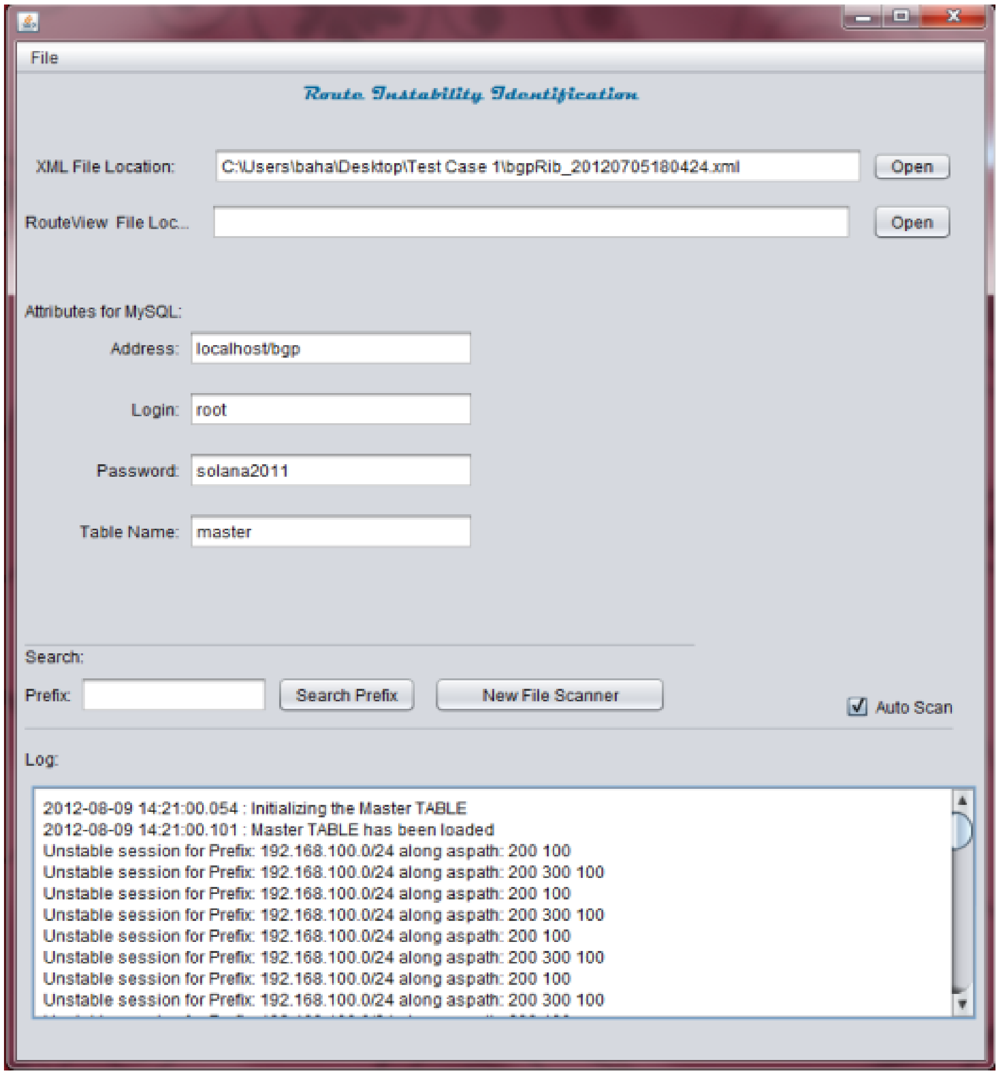
Database Table:



prefix	peerID	origin	MED	localP	NextHop	as_pathlength	as_path	timestamp	num_draws	num_updates	routeStatus
192.168.100.0/24	10.10.10.10	100	0	100	192.169.101.11	2	200 100	705184924	0	79	ADDED
192.168.200.0/24	10.10.10.10	200	0	100	192.169.101.11	1	200	705180424	0	0	static
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 4.10: Database Table converted from XML BGP Data

Output:



XML File Location: C:\Users\baha\Desktop\Test Case 1\bgpRib_20120705180424.xml

RouteView File Loc...

Attributes for MySQL:

Address: localhost/bgp

Login: root

Password: solana2011

Table Name: master

Search:

Prefix:

Log:

```
2012-08-09 14:21:00.054 : Initializing the Master TABLE
2012-08-09 14:21:00.101 : Master TABLE has been loaded
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 100
```

Figure 4.11: Program output of Test Case One

4.7.2: Test Case Two

The policies and the AS level network topology of this test case is almost the same as test case one. In this test case we create four ASes AS 100, 200, 300 and 400 using simulator. The destination network or the source network for which we are observing route oscillation is located in AS 100. AS 100 is established EBGP peer to all other ASes and AS 200, 300 and 400 also established EBGP peer among each other according to the figure 4.13. Here we manipulate the local preference attributes to setup the case.

The local policy for ASes are :

AS 200: Rank of path {200 300 100} < {200 100} (We set local preference for AS path {300 100} is 200 and keep others default.)

AS 300: Rank of path {300 400 100} < {300 100} (We set local preference for AS path {400 100} is 200 and keep others default.)

AS 400: Rank of path {400 200 100} < {400 100} (We set local preference for AS path {200 100} is 200 and keep others default.)

The lower rank means higher preference. Another AS 65000 has constructed using physical routers. Here R5 is a cisco 2811 Router and another router is Sonala Network Device. An EBGP peer has established between this physical AS 65000 with AS 200, which is constructed using the simulator.

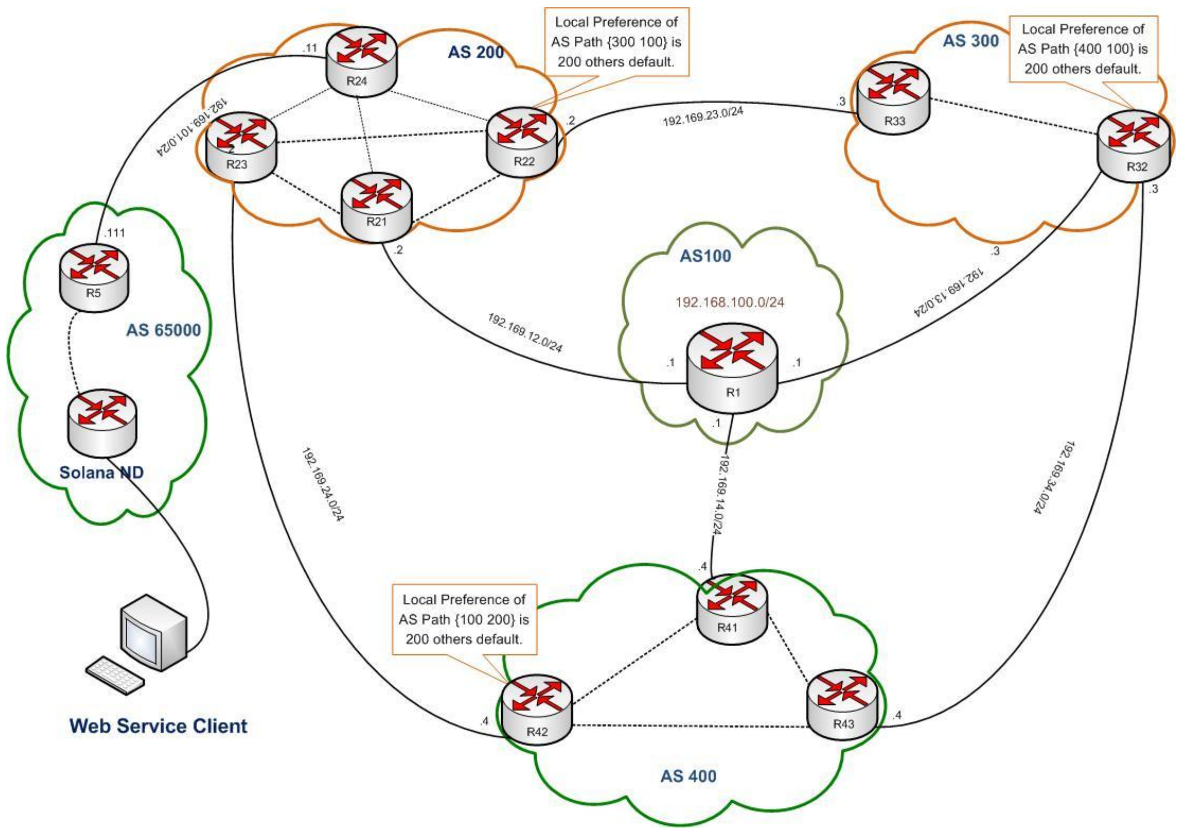


Figure 4.12: Test Case Two

After completing all the configuration according to the topology as above we start collecting BGP data and test them using the tool. BGP main table and updates collected according to the step by step test methodology explained in Appendix A has shown below.

```
<asNumber="65000">
 <Peer>
  <peerID>10.10.10.10</peerID>
   <Route>
    <prefix>192.168.20.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>200 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
   <Route>
    <prefix>192.168.30.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>200 300 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
   <Route>
    <prefix>192.168.40.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>200 400 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
```

```
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>200 300 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
 </Peer>
</ns:autonomousSystem>
```

Figure 4.13: BGP RIB of Test Case Two

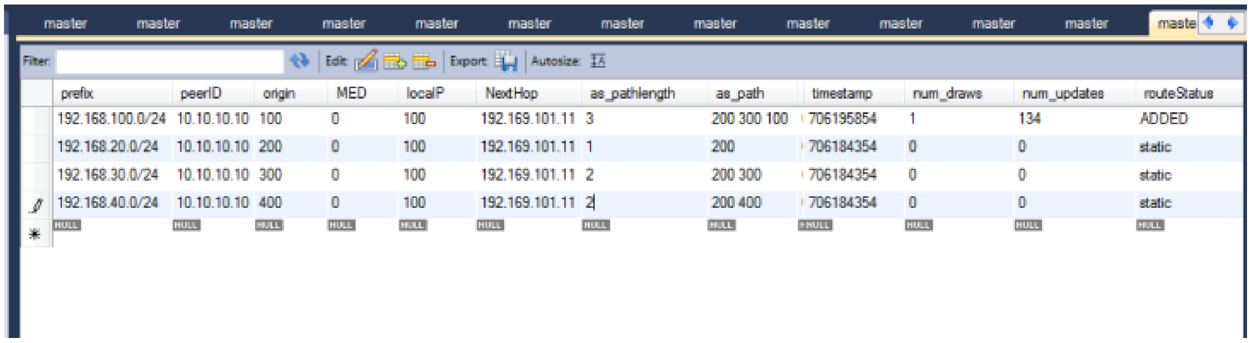
```
<Peer>
  <peerID>10.10.10.10</peerID>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <routeStatus>WITHDRAWN</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>200 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>200 300 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
```

```
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>200 300 400 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>200 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>200 300 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
```

```
 </nexthop>
 <multiexit>
  <multiExit>0</multiExit>
 </multiexit>
 <localpref>
  <localPref>100</localPref>
 </localpref>
 <routeStatus>ADDED</routeStatus>
</Route>
<Route>
 <prefix>192.168.100.0/24</prefix>
 <origin>
  <originId>IGP</originId>
 </origin>
 <aspath>
  <pathSegment>
   <path>200 300 400 100 </path>
  </pathSegment>
 </aspath>
 <nexthop>
  <nextHop>192.169.101.11</nextHop>
 </nexthop>
 <multiexit>
  <multiExit>0</multiExit>
 </multiexit>
 <localpref>
  <localPref>100</localPref>
 </localpref>
 <routeStatus>ADDED</routeStatus>
</Route>
<Route>
 <prefix>192.168.100.0/24</prefix>
 <origin>
  <originId>IGP</originId>
 </origin>
 <aspath>
  <pathSegment>
   <path>200 100 </path>
  </pathSegment>
 </aspath>
 <nexthop>
  <nextHop>192.169.101.11</nextHop>
 </nexthop>
 <multiexit>
  <multiExit>0</multiExit>
 </multiexit>
 <localpref>
  <localPref>100</localPref>
 </localpref>
 <routeStatus>ADDED</routeStatus>
```

Figure 4.14: BGP Sample Updates of Test Case Two

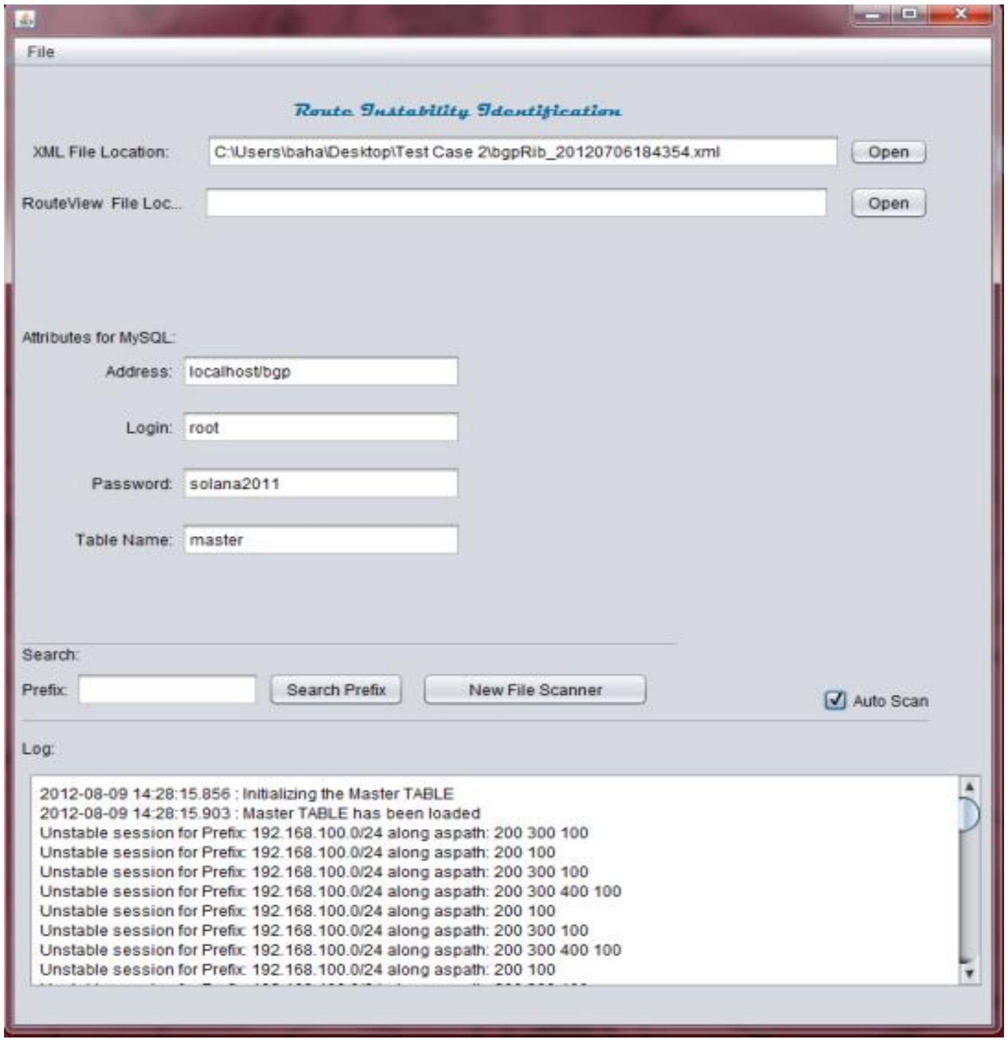
Database Table:



prefix	peerID	origin	MED	localP	NextHop	as_pathlength	as_path	timestamp	num_draws	num_updates	routeStatus
192.168.100.0/24	10.10.10.10	100	0	100	192.169.101.11	3	200 300 100	706195854	1	134	ADDED
192.168.20.0/24	10.10.10.10	200	0	100	192.169.101.11	1	200	706184354	0	0	static
192.168.30.0/24	10.10.10.10	300	0	100	192.169.101.11	2	200 300	706184354	0	0	static
192.168.40.0/24	10.10.10.10	400	0	100	192.169.101.11	2	200 400	706184354	0	0	static
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 4.15: Database Table for Test Case Two

Output



Route Instability Identification

XML File Location: C:\Users\baha\Desktop\Test Case 2\bgpRib_20120706184354.xml

RouteView File Loc...

Attributes for MySQL:

Address: localhost/bgp

Login: root

Password: solana2011

Table Name: master

Search:

Prefix:

[✓] Auto Scan

Log:

```
2012-08-09 14:28:15.856 : Initializing the Master TABLE
2012-08-09 14:28:15.903 : Master TABLE has been loaded
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 400 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 400 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 100
```

Figure 4.16: Program Output of Test Case Two

4.7.3: Test Case Three With Different Next Hop

In this test case we create total four ASes AS 65000, 100, 300 and 400 using simulator and Solana ND. The destination network or the source network for which we are observing route oscillation is located in AS 100. AS 100 is established EBGP peer to all other ASes and AS 65000, 300 and 400 also established EBGP peer among each other.

The local policy for ASes are :

AS 65000: Rank of path {65000 300 100} < {65000 100}

(We set local preference for AS path {300 100} is 200 and keep others default.)

AS 300: Rank of path {300 400 100} < {300 100}

(We set local preference for AS path {400 100} is 200 and keep others default.)

AS 400: Rank of path {400 65000 100} < {400 100}

(We set local preference for AS path {65000 100} is 200 and keep others default.)

The lower rank means higher preference. In AS 65000 Router R2 is a simulated router and another router is Solana Network Device (ND). An IBGP peer has established between this physical router Solana ND with simulated Router R2.

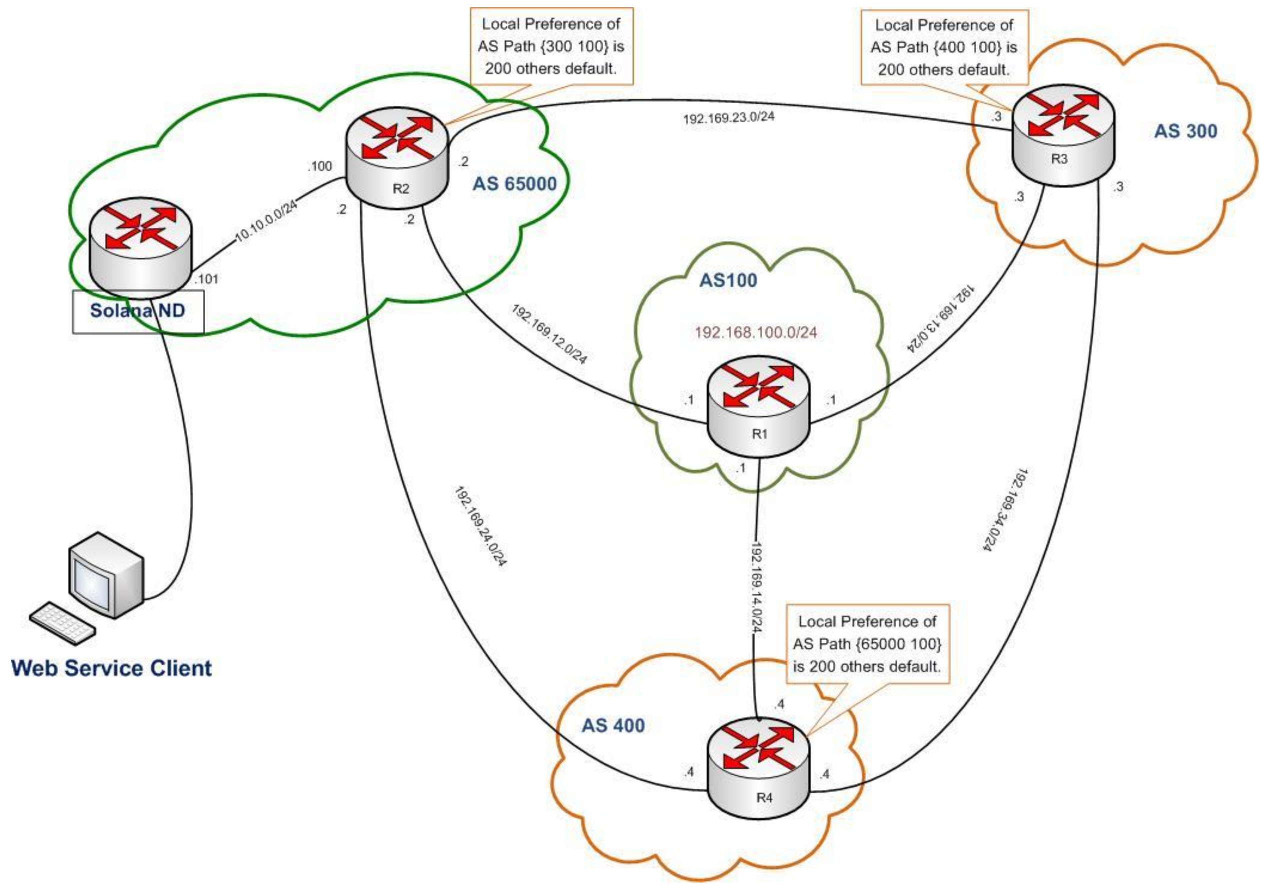

Local Preference of AS Path {300 100} is 200 others default.
Local Preference of AS Path {400 100} is 200 others default.
Local Preference of AS Path {65000 100} is 200 others default.
AS 65000
AS 300
AS100
AS 400
192.168.100.0/24
192.169.23.0/24
192.169.12.0/24
192.169.13.0/24
192.169.24.0/24
192.169.14.0/24
192.169.34.0/24
10.10.0.0/24
R1
R2
R3
R4
Solana ND
Web Service Client

Figure 4.17: Topology for Test Case Three manipulating Local Preference attribute

```
<asNumber="65000">
 <Peer>
  <peerID>192.168.200.2</peerID>
   <Route>
    <prefix>192.168.40.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>400 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.24.4</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>300 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.23.3</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>200</localPref>
    </localpref>
   </Route>
   <Route>
    <prefix>192.168.200.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <nexthop>
     <nextHop>10.10.0.100</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
 </Peer>
</ns:autonomousSystem>
```

Figure 4.18: BGP RIB for Test Case Three

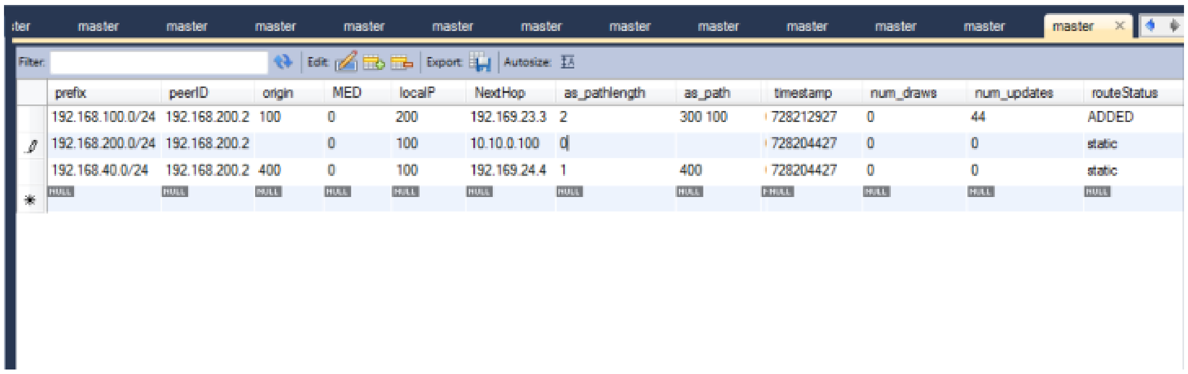
Updates

```
<Peer>
  <peerID>192.168.200.2</peerID>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>300 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.23.3</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>200</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.12.1</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
```

```
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>300 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.23.3</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>200</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.12.1</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>300 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.23.3</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>200</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.12.1</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
```

Figure 4.19: BGP Sample Update for Test Case Three

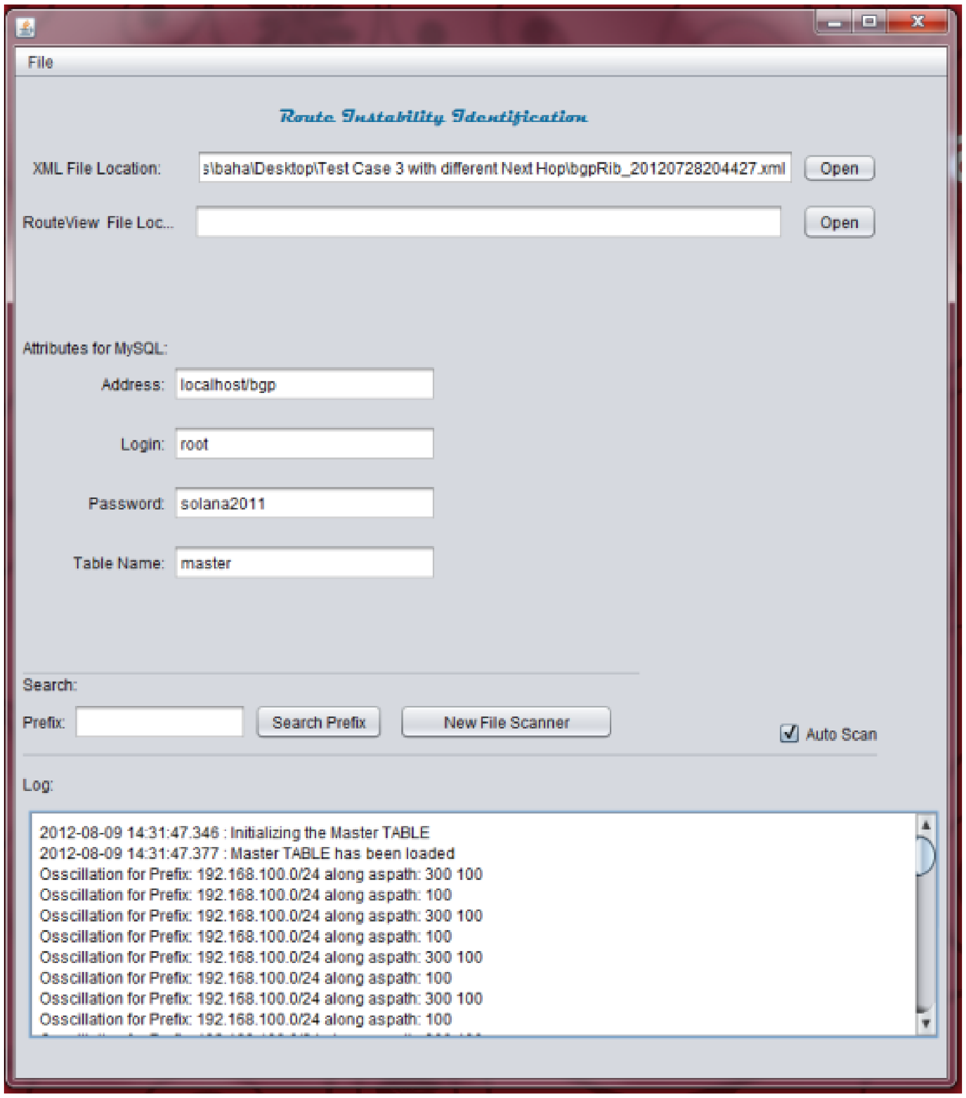
Database Table



prefix	peerID	origin	MED	localP	NextHop	as_pathlength	as_path	timestamp	num_draws	num_updates	routeStatus
192.168.100.0/24	192.168.200.2	100	0	200	192.169.23.3	2	300 100	728212927	0	44	ADDED
192.168.200.0/24	192.168.200.2		0	100	10.10.0.100	0		728204427	0	0	static
192.168.40.0/24	192.168.200.2	400	0	100	192.169.24.4	1	400	728204427	0	0	static
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 4.20: Database Table for Test Case Three

Output:



XML File Location: s\baha\Desktop\Test Case 3 with different Next Hop\bgpRib_20120728204427.xml

RouteView File Loc...

Attributes for MySQL:

Address: localhost/bgp

Login: root

Password: solana2011

Table Name: master

Search:

Prefix:

☑ Auto Scan

Log:

```
2012-08-09 14:31:47.346 : Initializing the Master TABLE
2012-08-09 14:31:47.377 : Master TABLE has been loaded
Osscillation for Prefix: 192.168.100.0/24 along aspath: 300 100
Osscillation for Prefix: 192.168.100.0/24 along aspath: 100
Osscillation for Prefix: 192.168.100.0/24 along aspath: 300 100
Osscillation for Prefix: 192.168.100.0/24 along aspath: 100
Osscillation for Prefix: 192.168.100.0/24 along aspath: 300 100
Osscillation for Prefix: 192.168.100.0/24 along aspath: 100
Osscillation for Prefix: 192.168.100.0/24 along aspath: 300 100
Osscillation for Prefix: 192.168.100.0/24 along aspath: 100
```

Figure 4.21: Output of Test Case Three

4.7.4: Test Case Four for Non-determinism

In this test case we try to check the non-determinism behaviour of BGP topology, where a network may change its destination path unpredictably as discussed in section 3.1.2. Here we set up four ASes As 100, 200, 300 and 400 using BGP simulator. AS 100 advertises prefix192.168.100.0 and we try to observe the non-determinism behaviour from AS 200. We established EBGP peer between AS 200 with the physical AS 65000. The business relationship among the ASes are as follows:

- The AS 300 and AS 400 has established a peer relationship among them.
- The AS 300 and AS 200 has established provider customer relationship between them.
- The AS 400 and AS 100 has established provider customer relationship between them.
- The AS 200 and AS 100 has established backup provider customer relationship between them using appropriate community attribute to tolerate link failure. So route advertisement directly comes from AS 100 get lower preference then route advertisement comes from AS 300. As a result AS200 will take 200 300 400 100 as primary path and will take 200 100 as backup path. The backup path will take only if primary path has goes down and if primary path comes up the AS should move to the primary path again.

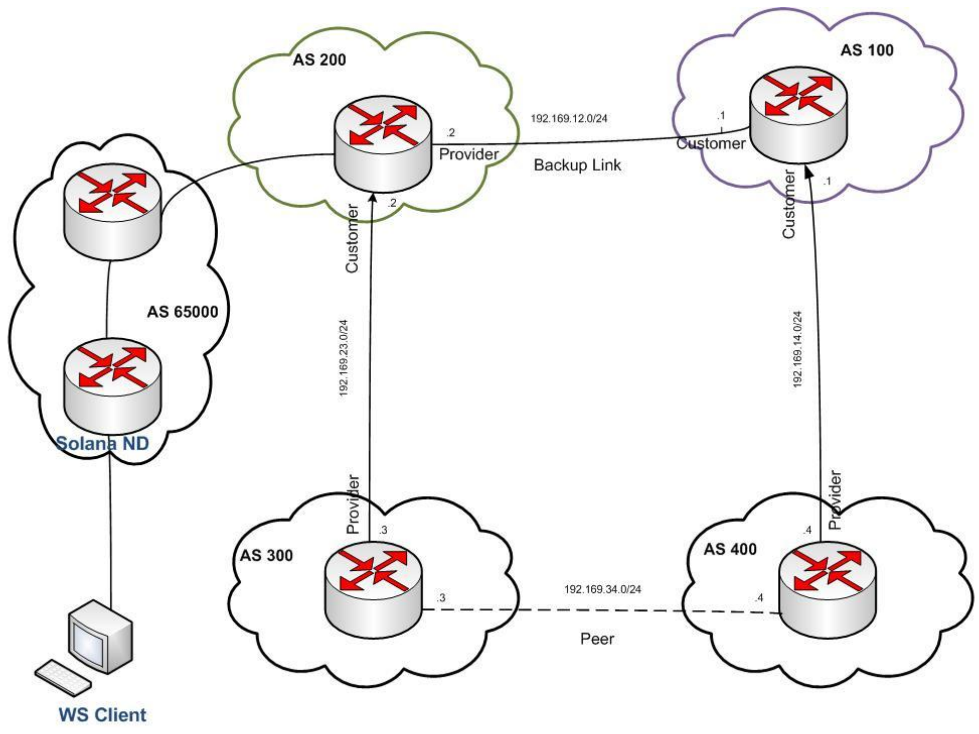


Figure 4.22: Topology for Test Case Four

Collected BGP RIB and updates are shown as below. These collected RIB and updates are tested using the tool and result of the tool has shown in figure 4.26.

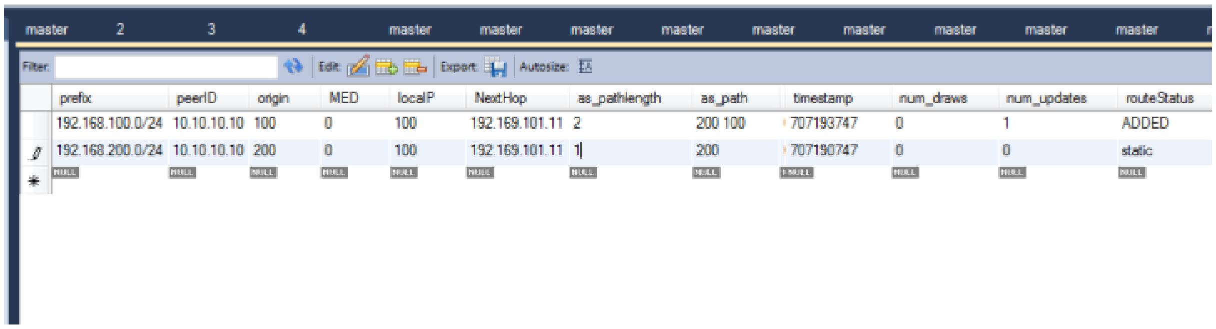
```
<asNumber="65000">
 <Peer>
  <peerID>10.10.10.10</peerID>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>200 300 400 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
   <Route>
    <prefix>192.168.200.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>200 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
 </Peer>
</ns:autonomousSystem>
```

Figure 4.0.23: BGP RIB of Test Case Four - Non-determinism

```
<asNumber="65000">
 <Peer>
  <peerID>10.10.10.10</peerID>
   <Route>
    <prefix>192.168.100.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>200 100 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.169.101.11</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
 </Peer>
</ns:autonomousSystem>
```

Figure 4.24: BGP Sample Updates of Test Case Four

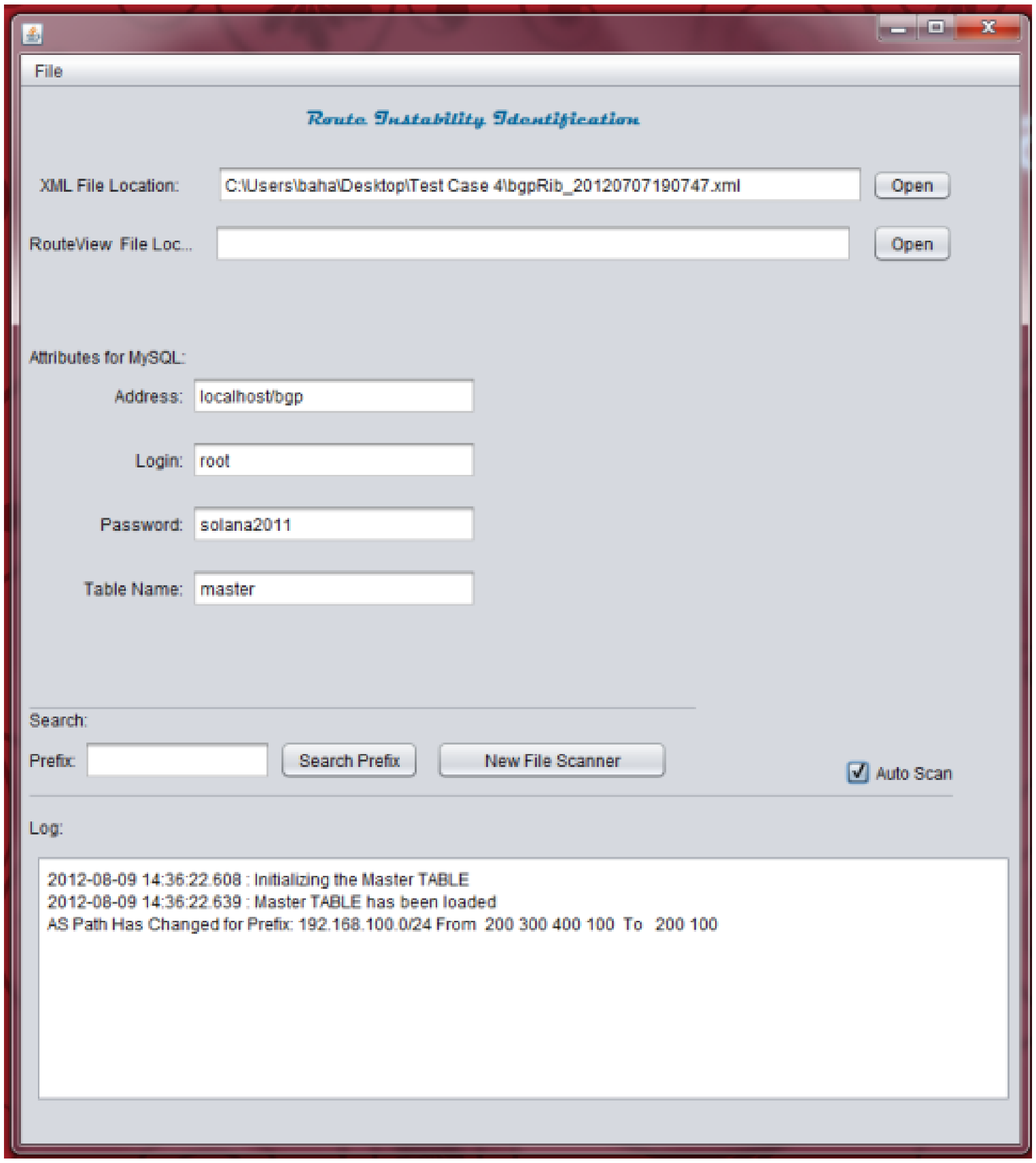
Database Table



prefix	peerID	origin	MED	localP	NextHop	as_pathlength	as_path	timestamp	num_draws	num_updates	routeStatus
192.168.100.0/24	10.10.10.10	100	0	100	192.169.101.11	2	200 100	707193747	0	1	ADDED
192.168.200.0/24	10.10.10.10	200	0	100	192.169.101.11	1	200	707190747	0	0	static
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 4.25: Database Table for Test Case Four

Output



XML File Location: C:\Users\baha\Desktop\Test Case 4\bgpRib_20120707190747.xml

Address: localhost/bgp

Login: root

Password: solana2011

Table Name: master

Auto Scan [✓]

```
2012-08-09 14:36:22.608 : Initializing the Master TABLE
2012-08-09 14:36:22.639 : Master TABLE has been loaded
AS Path Has Changed for Prefix: 192.168.100.0/24 From  200 300 400 100  To   200 100
```

Figure 4.26: Program Output of Test Case Four

4.8: Network Setup and Fault Test For Real World Data

In our study we test BGP policy induced topology disputes using the tool for both physical network (AS) and simulator generated test cases. We set up a private Autonomous System 65000 or a test network and connect that to Ryerson University Computing and Communications Services (CCS), a Ryerson university ISP (AS number is 26996) as shown in figure 4.25. We use both Cisco and Juniper routers to establish the network which helps us to understand details of the vendor implementations of an Autonomous System. To test the route oscillation in our test network we done that according to the step by step test methodology as we put a detailed description in appendix A.

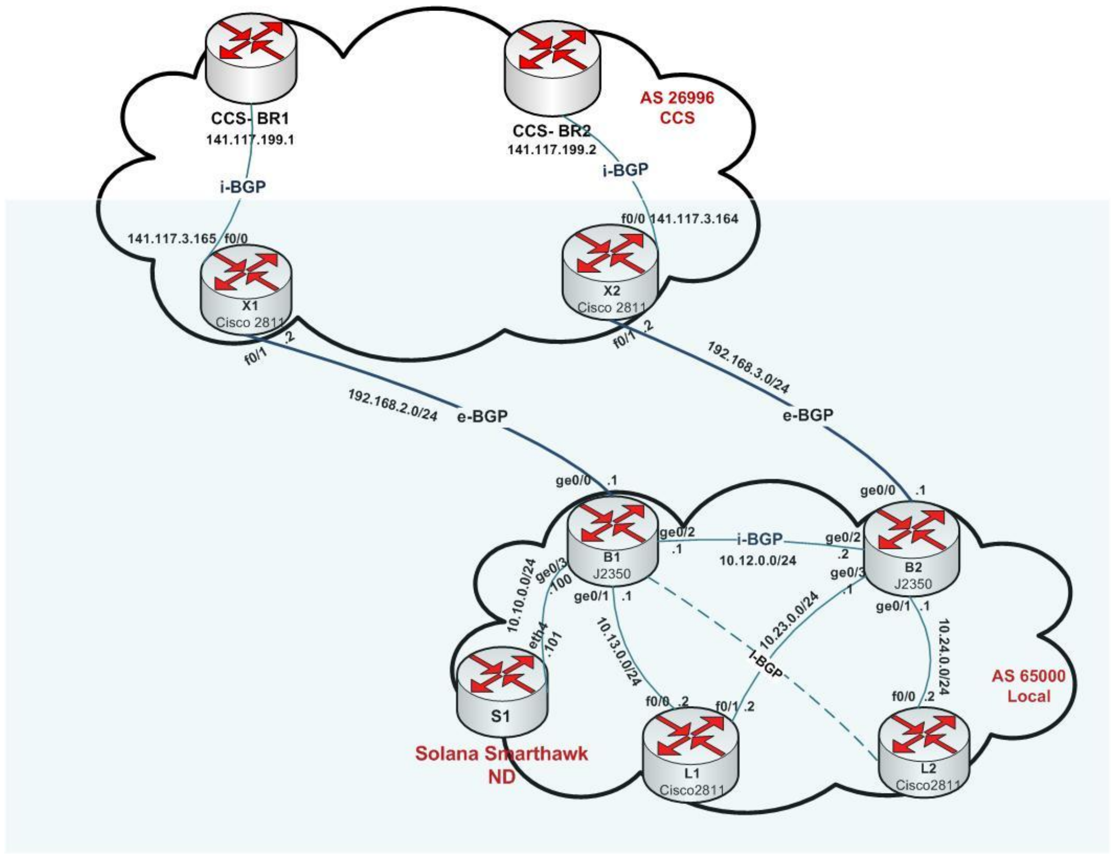


Figure 4.27 : Our Test Network with CCS

In this network topology or AS, S1 is the Solana Network ND and Cisco 2811 router is used to configure for two local routers L1 and L2. B1 and B2 are configured as two border routers using Juniper J2350 router. IBGP full mesh has established among all these four routers. Two other routers X1 and X2 has also configured as border router within Ryerson University Computing and Communications Services (CCS) network, AS number is 26996 and established IBGP connections with two CCS routers. Two EBGP connectivity has established between two border routers (B1 and B2) in our test AS with two other border routers (X1 and X2) in Ryerson University Computing and Communications Services (CCS) network, AS number is 26996. We use router B1 as the BGP speaker router for our test topology and connect that to Solana Smarthawk Network Device according to the step one of test methodology 1. About 400,000 routes and updates has collected according to the test methodology. These routes and updates has used as input to our BGP fault detection tool, below are some part of the Real world BGP RIB and Update files those we used as input in our tool for test. The output of the tool has also shown in figure 4.31.

```
<asNumber="65000">
 <Peer>
  <peerID>2.2.2.2</peerID>
   <Route>
    <prefix>1.0.0.0/24</prefix>
    <prefix>1.1.1.0/24</prefix>
    <prefix>1.2.3.0/24</prefix>
    <prefix>8.8.4.0/24</prefix>
    <prefix>8.8.8.0/24</prefix>
    <prefix>8.34.208.0/21</prefix>
    <prefix>8.34.216.0/21</prefix>
    <prefix>8.35.192.0/21</prefix>
    <prefix>8.35.200.0/21</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>26996 549 26677 15169 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.168.2.2</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
   <Route>
    <prefix>1.8.1.0/24</prefix>
    <prefix>1.8.8.0/24</prefix>
    <prefix>1.8.102.0/24</prefix>
    <prefix>1.8.150.0/24</prefix>
    <prefix>1.8.151.0/24</prefix>
    <prefix>1.8.152.0/24</prefix>
    <prefix>1.8.153.0/24</prefix>
    <prefix>1.8.240.0/24</prefix>
    <prefix>1.8.241.0/24</prefix>
    <prefix>1.8.242.0/24</prefix>
    <prefix>1.8.243.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>26996 549 26677 6509 11537 22388 7660 4641 4641 38345
</path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.168.2.2</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <aggregator>
     <aggregatorAS>38345</aggregatorAS>
     <aggregatorIP>202.45.188.209</aggregatorIP>
    </aggregator>
   </Route>
   <Route>
    <prefix>1.8.101.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>26996 549 26677 6509 11537 22388 7660 9264 7497 38345
</path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.168.2.2</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <aggregator>
     <aggregatorAS>38345</aggregatorAS>
     <aggregatorIP>1.8.101.254</aggregatorIP>
    </aggregator>
   </Route>
   <Route>
    <prefix>1.9.21.0/24</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>26996 549 26677 6509 11537 22388 7660 24287 24490 24514
</path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.168.2.2</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
   <Route>
    <prefix>1.51.0.0/16</prefix>
    <prefix>1.184.0.0/15</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <segmentType>AS_SEQUENCE</segmentType>
      <path>26996 549 26677 6509 20388 17579 23911 4538 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.168.2.2</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
   </Route>
```

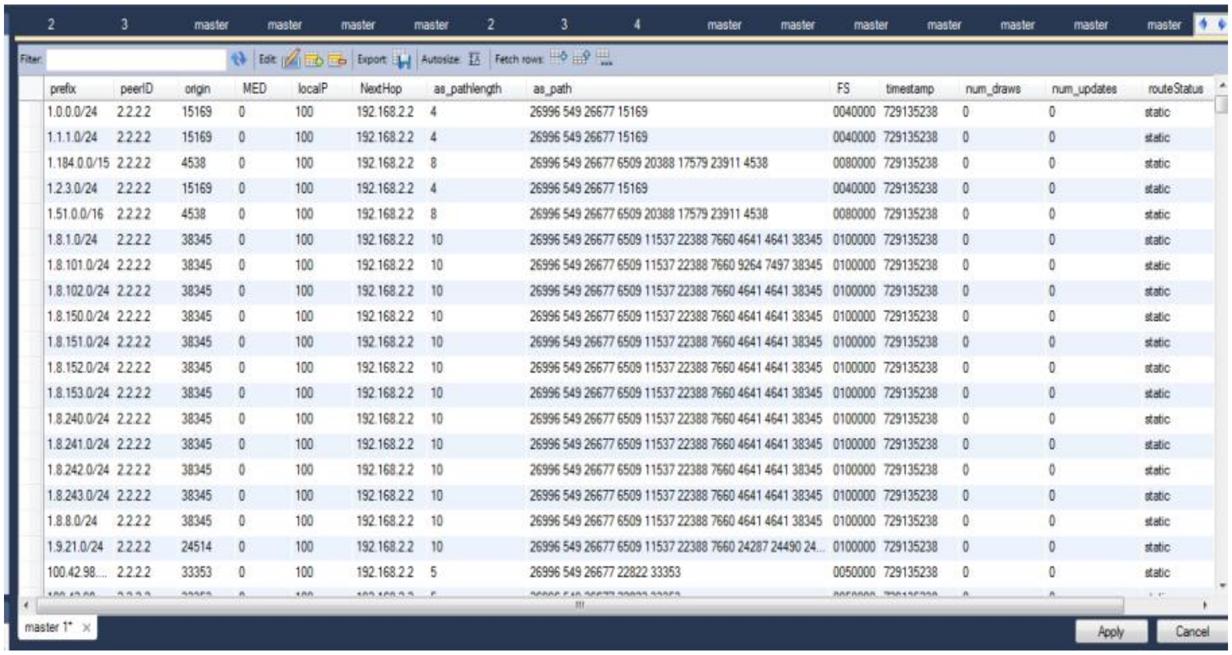
Figure 4.28: Sample BGP RIB for Real BGP Data

```
<Peer>
  <peerID>2.2.2.2</peerID>
   <Route>
    <prefix>187.141.66.0/28</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>26996 549 26677 6509 18592 8151 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.168.2.2</nextHop>
    </nexthop>
    <multiexit>
     <multiExit>0</multiExit>
    </multiexit>
    <localpref>
     <localPref>100</localPref>
    </localpref>
    <routeStatus>ADDED</routeStatus>
   </Route>
   <Route>
    <prefix>187.141.130.144/28</prefix>
    <origin>
     <originId>IGP</originId>
    </origin>
    <aspath>
     <pathSegment>
      <path>26996 549 26677 6509 18592 8151 </path>
     </pathSegment>
    </aspath>
    <nexthop>
     <nextHop>192.168.2.2</nextHop>
    </nexthop>
```

```
     <multiexit>
      <multiExit>0</multiExit>
     </multiexit>
     <localpref>
      <localPref>100</localPref>
     </localpref>
     <routeStatus>ADDED</routeStatus>
    </Route>
    <Route>
     <prefix>190.69.154.8/29</prefix>
     <origin>
      <originId>IGP</originId>
     </origin>
     <aspath>
      <pathSegment>
       <path>26996 549 26677 6509 20965 27750 27817 </path>
      </pathSegment>
     </aspath>
     <nexthop>
      <nextHop>192.168.2.2</nextHop>
     </nexthop>
     <multiexit>
      <multiExit>0</multiExit>
     </multiexit>
     <localpref>
      <localPref>100</localPref>
     </localpref>
     <routeStatus>ADDED</routeStatus>
    </Route>
    <Route>
     <prefix>190.69.154.8/29</prefix>
     <origin>
      <originId>IGP</originId>
     </origin>
     <aspath>
      <pathSegment>
       <path>26996 549 26677 6509 27750 27817 </path>
      </pathSegment>
     </aspath>
     <nexthop>
      <nextHop>192.168.2.2</nextHop>
     </nexthop>
     <multiexit>
      <multiExit>0</multiExit>
     </multiexit>
     <localpref>
      <localPref>100</localPref>
     </localpref>
     <routeStatus>ADDED</routeStatus>
    </Route>
```

Figure 4.29: Sample BGP Updates for Real Data

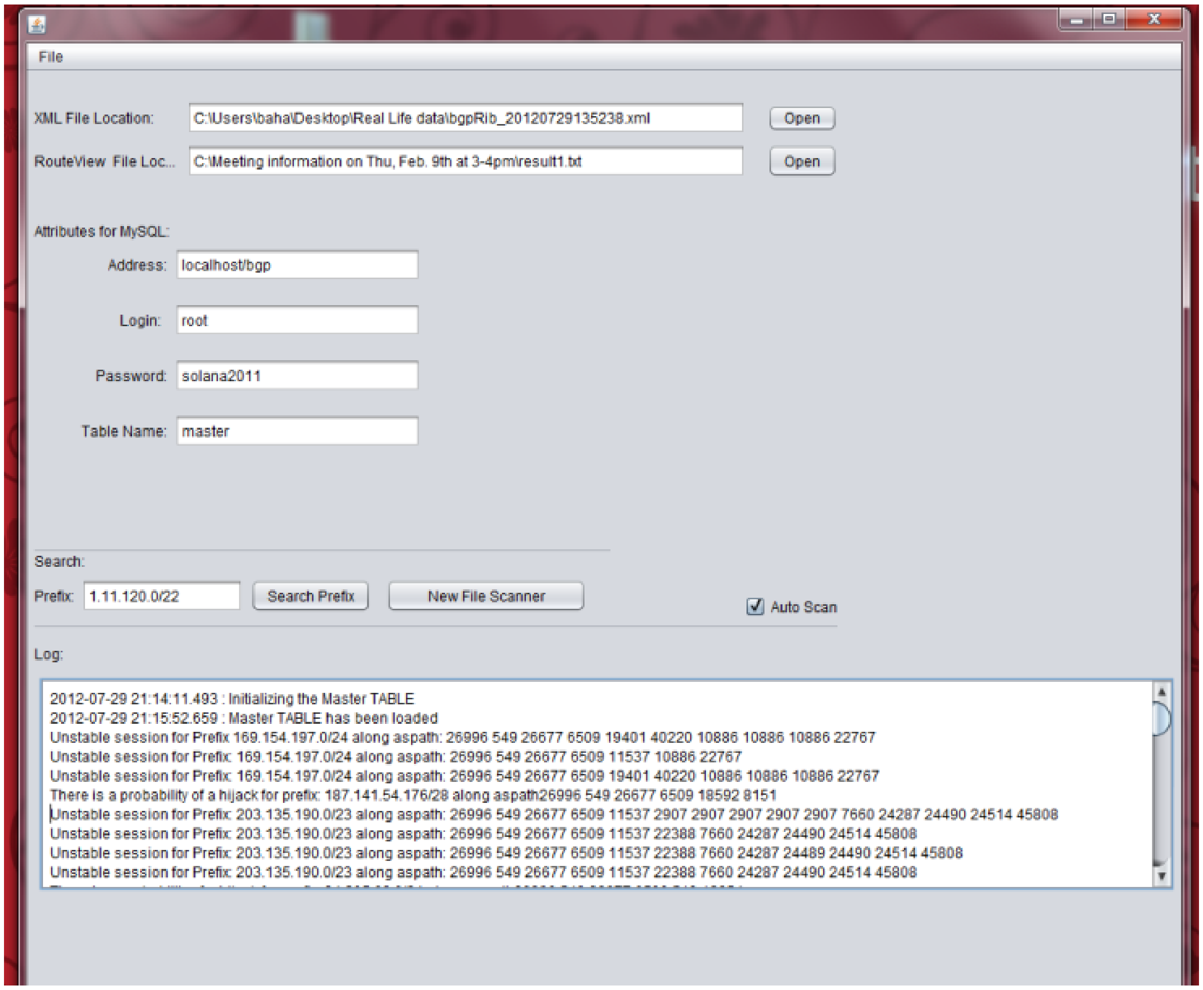
Database Table



prefix	peerID	origin	MED	localP	NextHop	as_pathlength	as_path	FS	timestamp	num_draws	num_updates	routeStatus
1.0.0.0/24	2.2.2.2	15169	0	100	192.168.2.2	4	26996 549 26677 15169	0040000	729135238	0	0	static
1.1.1.0/24	2.2.2.2	15169	0	100	192.168.2.2	4	26996 549 26677 15169	0040000	729135238	0	0	static
1.184.0.0/15	2.2.2.2	4538	0	100	192.168.2.2	8	26996 549 26677 6509 20388 17579 23911 4538	0080000	729135238	0	0	static
1.2.3.0/24	2.2.2.2	15169	0	100	192.168.2.2	4	26996 549 26677 15169	0040000	729135238	0	0	static
1.51.0.0/16	2.2.2.2	4538	0	100	192.168.2.2	8	26996 549 26677 6509 20388 17579 23911 4538	0080000	729135238	0	0	static
1.8.1.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 4641 4641 38345	0100000	729135238	0	0	static
1.8.101.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 9264 7497 38345	0100000	729135238	0	0	static
1.8.102.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 4641 4641 38345	0100000	729135238	0	0	static
1.8.150.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 4641 4641 38345	0100000	729135238	0	0	static
1.8.151.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 4641 4641 38345	0100000	729135238	0	0	static
1.8.152.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 4641 4641 38345	0100000	729135238	0	0	static
1.8.153.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 4641 4641 38345	0100000	729135238	0	0	static
1.8.240.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 4641 4641 38345	0100000	729135238	0	0	static
1.8.241.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 4641 4641 38345	0100000	729135238	0	0	static
1.8.242.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 4641 4641 38345	0100000	729135238	0	0	static
1.8.243.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 4641 4641 38345	0100000	729135238	0	0	static
1.8.8.0/24	2.2.2.2	38345	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 4641 4641 38345	0100000	729135238	0	0	static
1.9.21.0/24	2.2.2.2	24514	0	100	192.168.2.2	10	26996 549 26677 6509 11537 22388 7660 24287 24490 24...	0100000	729135238	0	0	static
100.42.98....	2.2.2.2	33353	0	100	192.168.2.2	5	26996 549 26677 22822 33353	0050000	729135238	0	0	static

Figure 4.30: Sample BGP Database Table

Output



XML File Location: C:\Users\baha\Desktop\Real Life data\bgpRib_20120729135238.xml

RouteView File Loc... C:\Meeting information on Thu, Feb. 9th at 3-4pm\result1.txt

Attributes for MySQL:

Address: localhost/bgp

Login: root

Password: solana2011

Table Name: master

Search:

Prefix: 1.11.120.0/22

[✓ Auto Scan]

Log:

```
2012-07-29 21:14:11.493 : Initializing the Master TABLE
2012-07-29 21:15:52.659 : Master TABLE has been loaded
Unstable session for Prefix 169.154.197.0/24 along aspath: 26996 549 26677 6509 19401 40220 10886 10886 10886 22767
Unstable session for Prefix: 169.154.197.0/24 along aspath: 26996 549 26677 6509 11537 10886 22767
Unstable session for Prefix: 169.154.197.0/24 along aspath: 26996 549 26677 6509 19401 40220 10886 10886 10886 22767
There is a probability of a hijack for prefix: 187.141.54.176/28 along aspath26996 549 26677 6509 18592 8151
Unstable session for Prefix: 203.135.190.0/23 along aspath: 26996 549 26677 6509 11537 2907 2907 2907 2907 2907 7660 24287 24490 24514 45808
Unstable session for Prefix: 203.135.190.0/23 along aspath: 26996 549 26677 6509 11537 22388 7660 24287 24490 24514 45808
Unstable session for Prefix: 203.135.190.0/23 along aspath: 26996 549 26677 6509 11537 22388 7660 24287 24489 24490 24514 45808
Unstable session for Prefix: 203.135.190.0/23 along aspath: 26996 549 26677 6509 11537 22388 7660 24287 24490 24514 45808
```

Figure 4.31: Sample Output of Real Data

Chapter Five

Conclusion and Future Work

Today's Internet has been constructed upon the deployment of Border Gateway Protocol (BGP), that enables service providers to establish routing among each other and maintain the global reachability. BGP runs a global distribution network, within that nodes are the BGP routers and links are the BGP sessions established among BGP neighbouring routers. The performance of BGP is very vital for a healthy and efficient global routing and faults in BGP routing may disrupt large section of the Internet. On the other hand, ASes would like to set policies according to their own business policies and keep these policies private. These local policies may create global conflicts and lead to route oscillation or non-determinism. The main contribution of this thesis are as follows:

- An extensive study on BGP policy conflicts and their available proposed solution.
- An algorithm and its implementation to detect route instability within an AS and report that to network administrator so that he/she can respond quickly. This tool has been tested by real life data collected through our test network and test cases developed by BGP simulator.
- A new dynamic algorithm to resolve Inter-Domain policy conflicts or route oscillation. The proposed algorithm reduces the memory overhead due to maintain path history. It uses local information and does not need to send history in the update message. Therefore it reduces the communication and memory overhead compare to other solutions.
- The implementation of the proposed algorithm using SimBGP simulator. Three test cases, their topologies and configurations that we run in the simulator in the test-bed. Finally, we show the results in automatically resolving the route oscillation in every test topology.

This study can be extended as follows as our future work:

- Route hijacking is another burning issue now a days. So it would be a wonderful work if it is possible to develop and implement the identification of route hijacking and integrating that with this tool. For this we need to collect and analyze BGP real life data from another source like Route View. This work can also be extend by giving a visualization of the AS path links.

- From the view point of our proposed algorithm to resolve the Inter-Domain policy conflict we can implement and test this algorithm using both real life and simulated data. Performance analysis of this algorithm could be another very good addition to see how quickly this proposed algorithm adapts.

Instead of resolving conflicts automatically, the conflicts can be identified by the tool as we proposed and developed. In this case the Network administrator could be manually fixed this problem according to the generated report by the tool. But if we can implement this Inter-Domain policy conflict algorithm, it will increase the efficiency and transparency of the protocol.

Appendix A

BGP Step-by-Step Fault Test Methodology

Today's internet is an intricate network of Autonomous Systems (AS) and Border Gateway Protocol (BGP) is the underlying protocol of this internet. Now this protocol is already in its fourth generation. It was first created in 1989 and codified in RFC 1105, but the protocol had to be updated to put up the genesis of new technologies and the improvements of routing performance. BGP-4 specified by RFC 4271, was adopted in 2006. RFC 4271 has superseded RFC 1771 and others, also has been amended several times to introduce modifications and corrections.

BGP offers optimized communications within the internet and selects the best path to the destination. It is highly responsive to recurrent network topology changes and can accommodate unconventional routing requirements. The persistent growth of the Internet cause many BGP operational challenges. Huge number of links regularly added and deleted, and the importance is directing traffic in the best route based on BGP policies. It's implementation vary considerably between vendors. That is why companies must test their networks or ASes for BGP route convergence to ensure loop free, oscillation free and identify route non-determinism to continue optimal performance.

This software application operates with the Solana Network Device and Solana Web Service Client. In this appendix we describe step by step test methodologies for Policy-Induced route oscillation and route non-determinism problems based on our proposed BGP tool.

Step by Step BGP Fault Detection Using Tool

Objective

This test will establish a BGP neighbour relationship with a BGP speaker router of test AS or test network, collect the BGP updates, process the updates, run the algorithm and take the decision about oscillation is there or not.

Step One

Establish an IBGP neighbour relationship between one BGP speaker router of test network and Solana network device (ND) using any one of the eight Ethernet port. Configure correct IP address and Autonomous System Number in Solana ND.

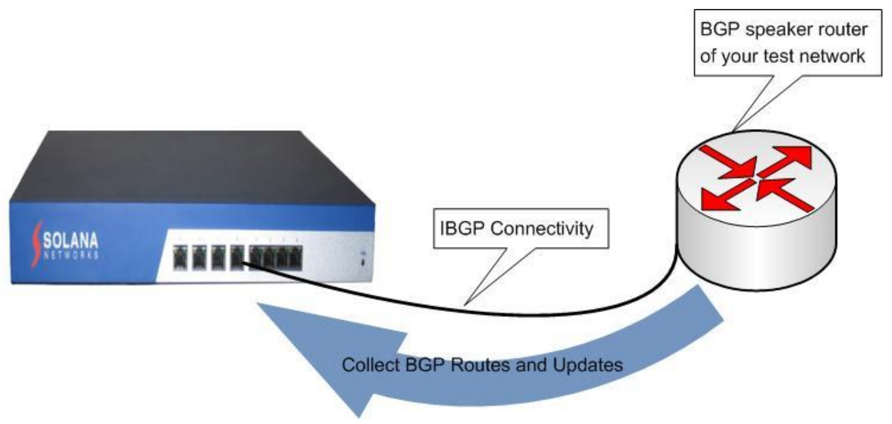

BGP speaker router of your test network
SOLANA NETWORKS
IBGP Connectivity
Collect BGP Routes and Updates

Figure A.1: BGP Speaker of Test Network and Solana ND Connection Setup

Step two

Configure a network connection with appropriate IP address between a Linux Red Hat 9 workstation and Solana network device using a switch.

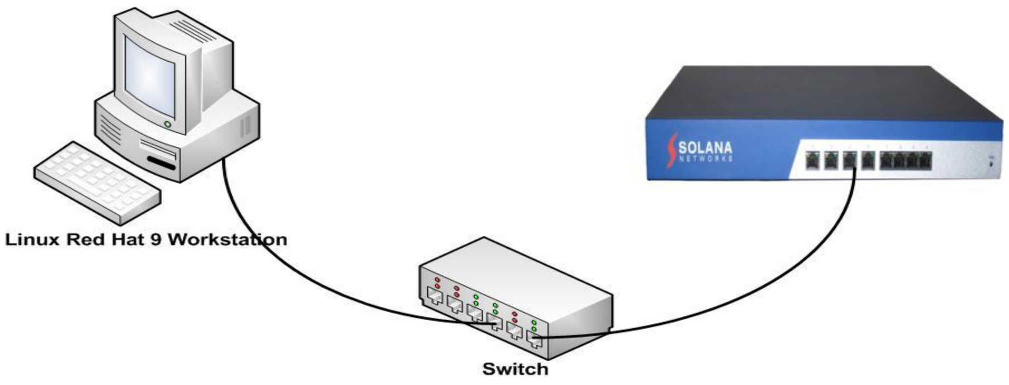


Figure A.2: Network Configuration of Solana ND and Red Hat Linux Server

Step Three

A sample *Makefile* is provided in the web service client C++ package to build the BGP client executable. Run this Makefile of the Web Service Client package at linux red hat 9 workstation. After successful execution connect the WS Client package to the Solana ND using the following command:

./mtomClient –g < IPAddress of Solana Network Device>

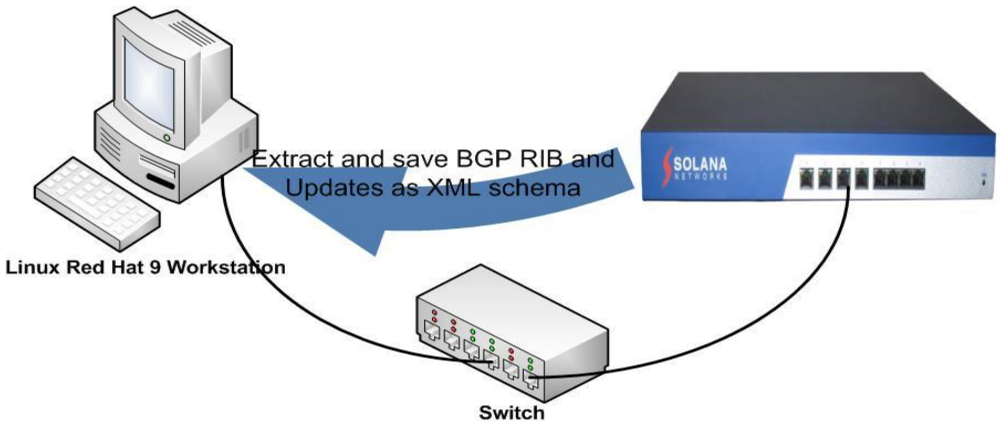

Extract and save BGP RIB and Updates as XML schema
SOLANA NETWORKS
Linux Red Hat 9 Workstation
Switch

Figure A.3: BGP Data extraction from Solana ND

The test client runs in a loop requesting BGP RIB data every 15 minutes to Solana ND. At first run, whole BGP RIB is requested and afterwards BGP updates arrived in the previous 15 minutes are retrieved. The 15 minutes loop time will have to be increased, if first BGP data retrieval takes longer than

15 minutes because BGP RIB may contains ~400,000 routes. A successful connection between the web service and Solana ND will extract and save BGP RIB with the file name of format bgpRib_yyyymmddhhmmss.xml as an XML schema for BGP RIB and updates data.

Step Four

Set up another connection from switch to a Windows workstation and configure appropriate IP address so that windows and Linux workstations can communicate each other.

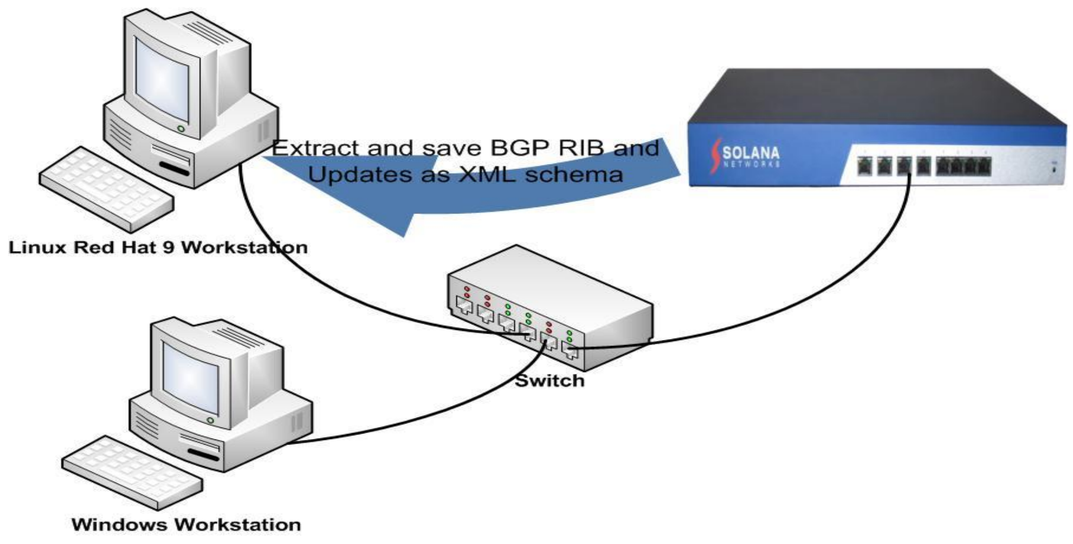

Extract and save BGP RIB and Updates as XML schema
SOLANA NETWORKS
Linux Red Hat 9 Workstation
Switch
Windows Workstation

Figure A.4: Network Setup Among Windows Workstation, Solana ND and Red Hat Linux Server

Step Five

Install MySQL and NetBeans IDE, Before installing NetBeans you need to install JDK 1.6 or above in the in the windows workstation machine. We use MySQL to maintain our database and NetBeans as development environment. Now open our BGP fault analysis project using NetBeans. This analysis software will use the BGP RIB and updates XML files as its input from Linux workstation, those we collect through Solana ND.

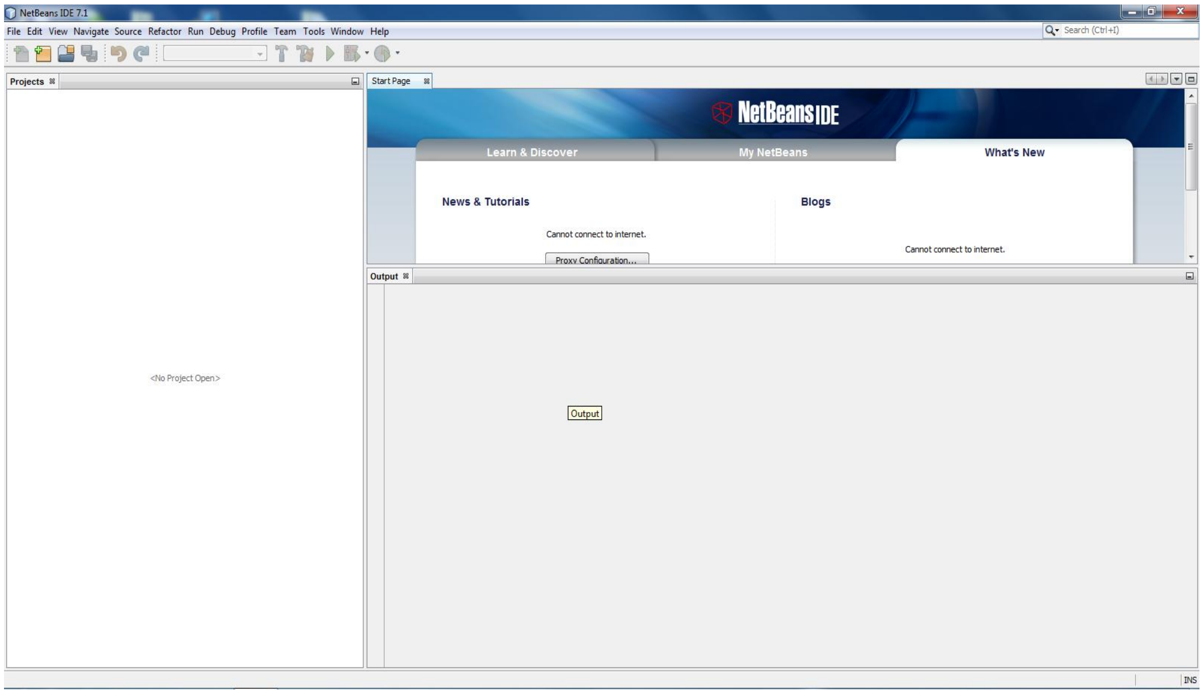
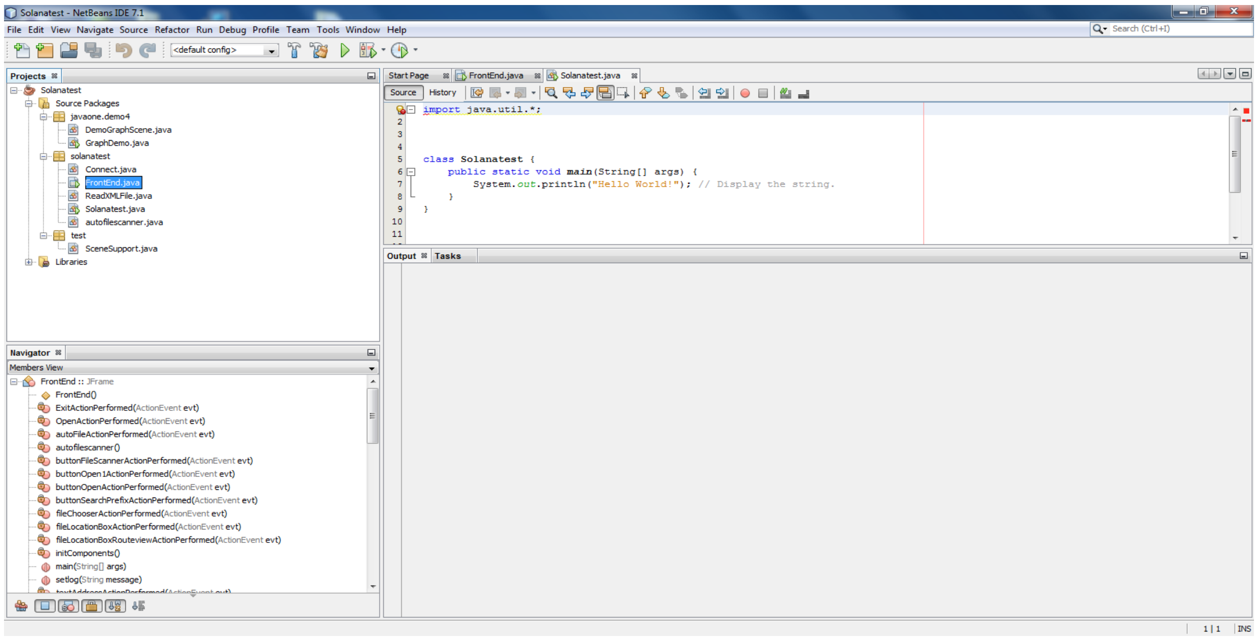


Figure A.5: NetBeans Java Development Environment

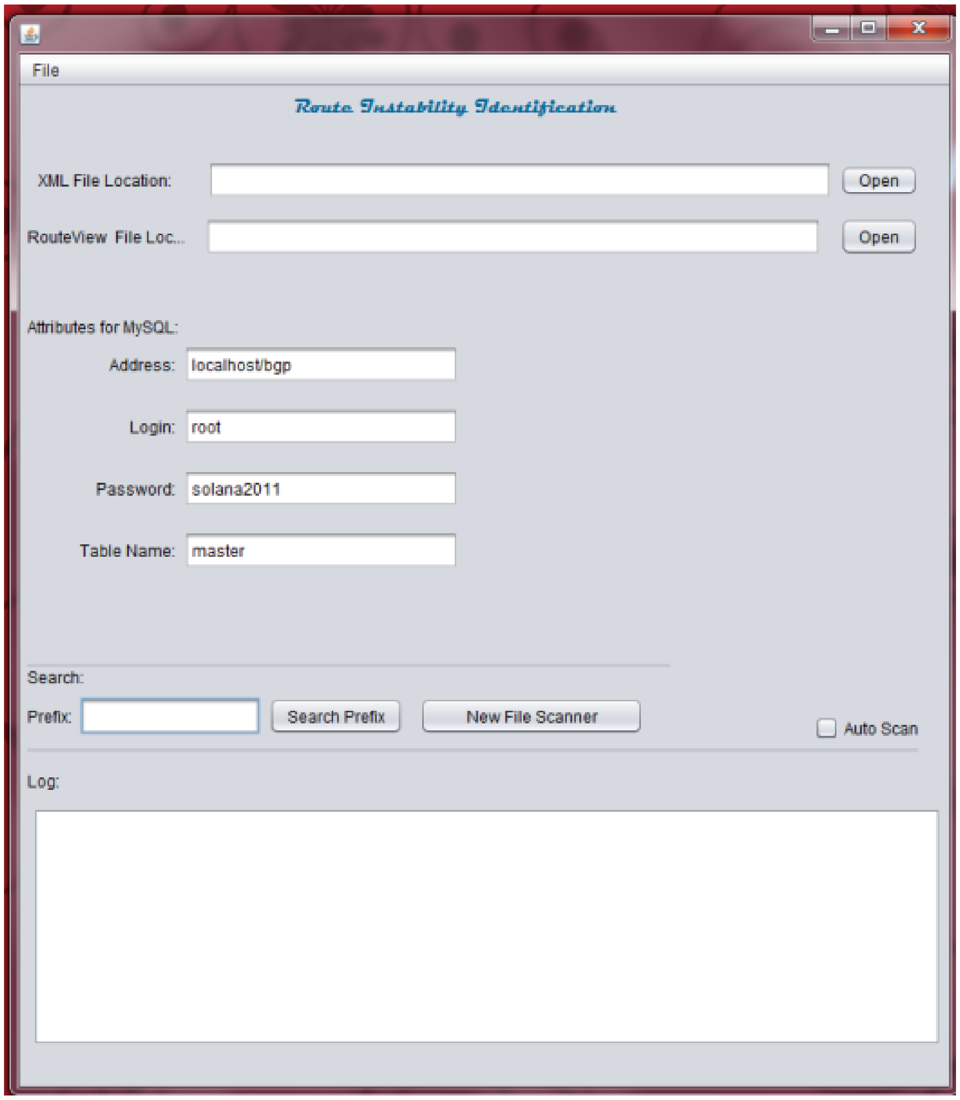

Solanatest - NetBeans IDE 7.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Search (Ctrl+I)
Projects
Solanatest
Source Packages
javaone.demo4
DemoGraphScene.java
GraphDemo.java
solanatest
Connect.java
FrontEnd.java
ReadXMLFile.java
Solanatest.java
autofilescanner.java
test
SceneSupport.java
Libraries
Start Page
FrontEnd.java
Solanatest.java
Source
History
Navigator
Members View
FrontEnd :: JFrame
FrontEnd()
ExitActionPerformed(ActionEvent evt)
OpenActionPerformed(ActionEvent evt)
autoFileActionPerformed(ActionEvent evt)
autofilescanner()
buttonFileScannerActionPerformed(ActionEvent evt)
buttonOpen1ActionPerformed(ActionEvent evt)
buttonOpenActionPerformed(ActionEvent evt)
buttonSearchPrefixActionPerformed(ActionEvent evt)
fileChooserActionPerformed(ActionEvent evt)
fileLocationBoxActionPerformed(ActionEvent evt)
fileLocationBoxRouteviewActionPerformed(ActionEvent evt)
initComponents()
main(String[] args)
setlog(String message)
Output
Tasks
1 | 1
INS

```
import java.util.*;


class Solanatest {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
```

Figure A.6: NetBeans Java Development Environment After opening The fault analysis software

Step Six

From the Project pannel select FontEnd.java and run the project, a new window will come out as below. Using the Open button select the XML file location, BGP RIB and updates.



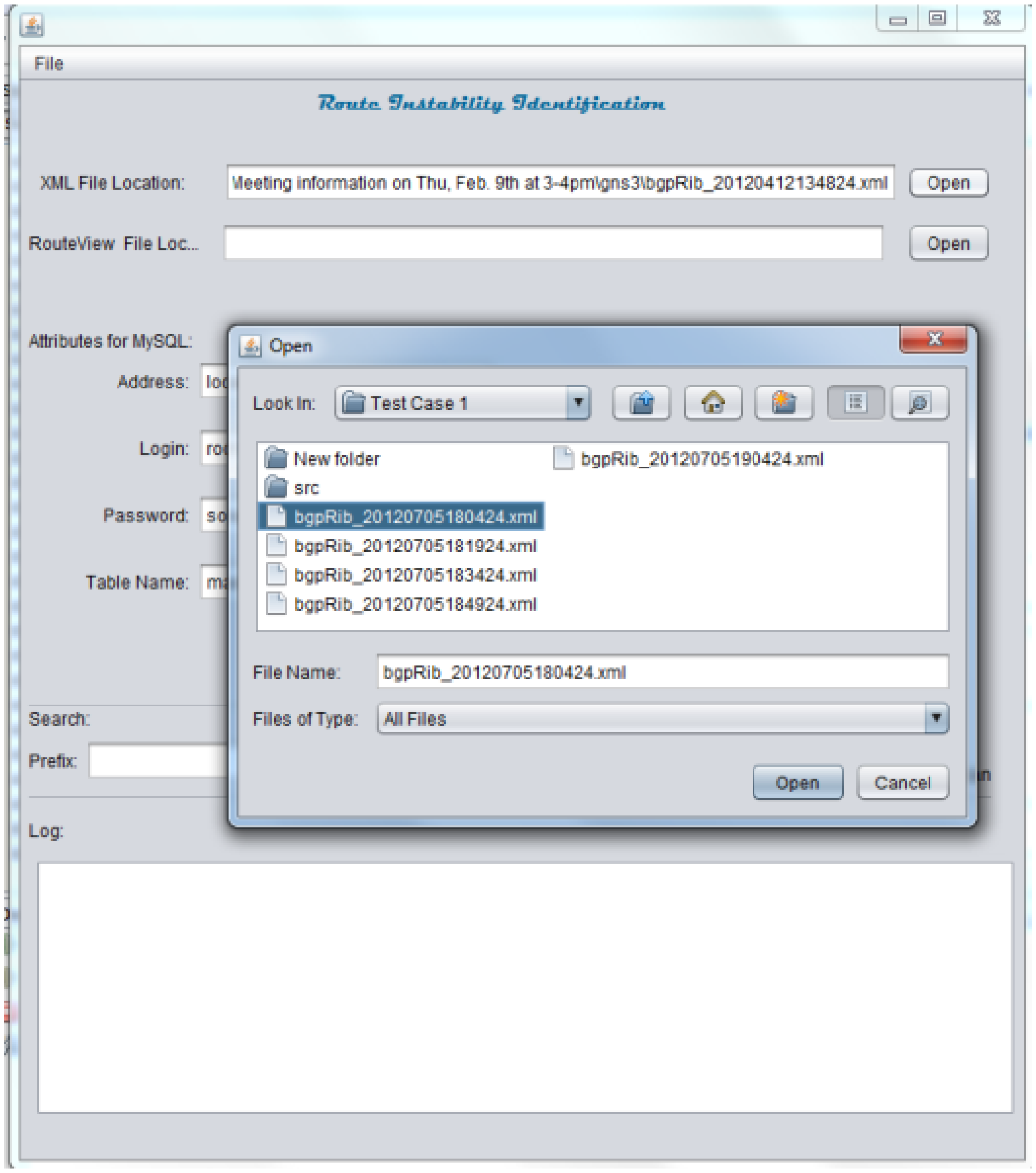
Address: localhost/bgp

Login: root

Password: solana2011

Table Name: master

Figure A.7: BGP fault analysis software



XML File Location: \Meeting information on Thu, Feb. 9th at 3-4pm\gns3\bgpRib_20120412134824.xml

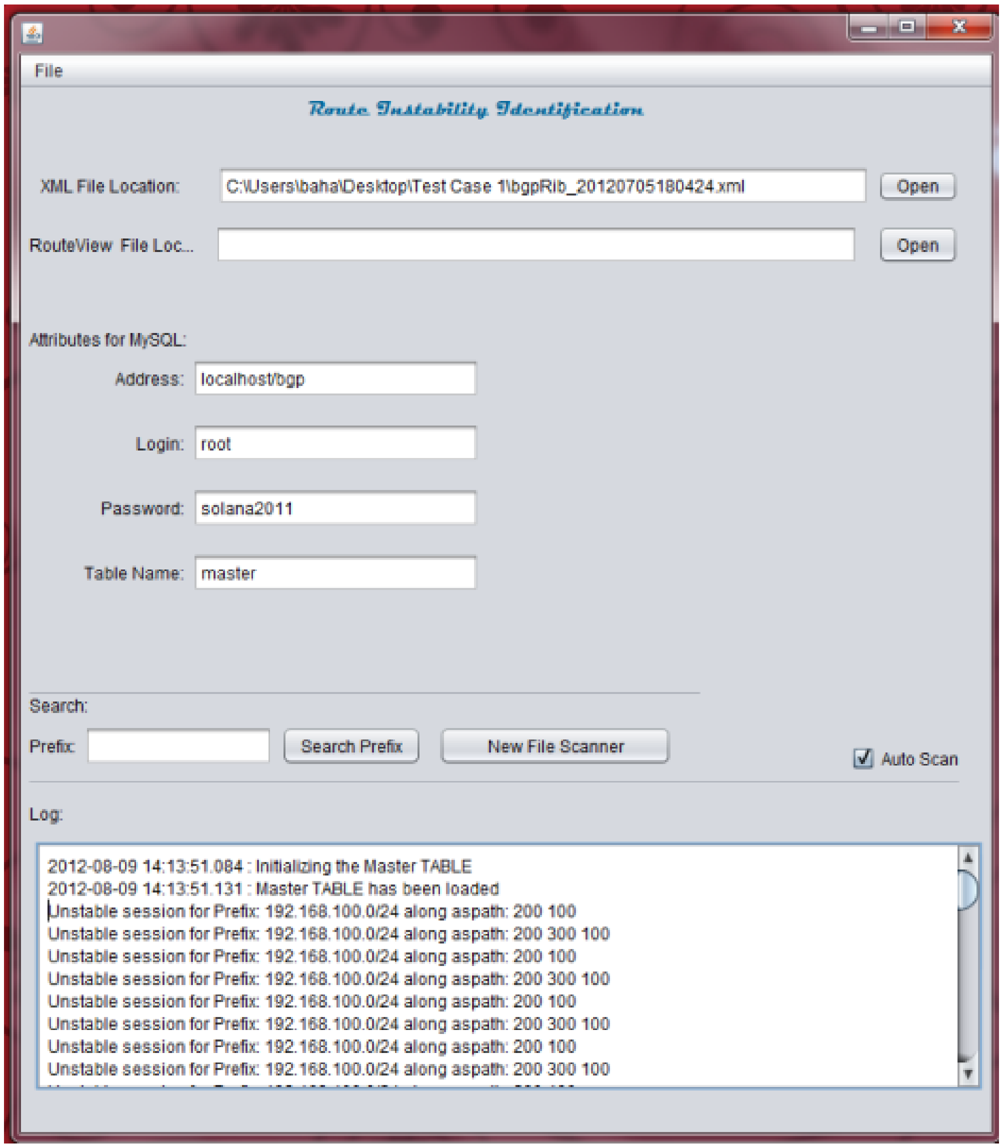
New folder
src
bgpRib_20120705180424.xml
bgpRib_20120705181924.xml
bgpRib_20120705183424.xml
bgpRib_20120705184924.xml
bgpRib_20120705190424.xml

File Name: bgpRib_20120705180424.xml

Figure A.8: BGP RIB and Updates Path Selection

Step Seven

Now press thw Auto Scan Button and the tool will show you the output, does there any Oscillation, unstable session or non-deterministic route based on our algorithm.



XML File Location: C:\Users\baha\Desktop\Test Case 1\bgpRib_20120705180424.xml

RouteView File Loc...

Attributes for MySQL:

Address: localhost/bgp

Login: root

Password: solana2011

Table Name: master

Search:

Prefix:

Log:

```
2012-08-09 14:13:51.084 : Initializing the Master TABLE
2012-08-09 14:13:51.131 : Master TABLE has been loaded
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 100
Unstable session for Prefix: 192.168.100.0/24 along aspath: 200 300 100
```

Figure A.9: Output of Route Oscillation

References

[1] Balon, S. and Leduc, G., BGP-aware IGP Link Weight Optimization in Presence of Route Reflectors. *IEEE INFOCOM, 2009*

[2] Basu, A., Ong, C., L., Rasala, A., Shepherd, F., B., Wilfong, G., Route Oscillation in I-BGP with Route Reflection. *in Proc. of the 2002 (SIGCOMM '02) conference on Applications, technologies, architectures, and protocols for computer communications.*

[3] Caesar, M., Subramanian, L., and Katz, R. H. Towards localizing root causes of BGP dynamics. *Tech. Rep. CSD-03-1292, UC Berkeley, November 2003*.

[4] Caeser, M., Rexford, J., BGP Routing Policies in ISP Networks. *IEEE Network, November/December 2005*.

[5] Chandra, R., Traina, P., BGP Communities Attribute. RFC 1997, August 1996.

[6] Chang, D.-F.,Govindan, R., and Heidemann, J. The temporal and topological characteristics of BGP path changes. In *Proc. IEEE ICNP* (November 2003).

[7] Chau, C.-K. (2006). Policy-based routing with non-strict preferences. In *Proc. ACM SIGCOMM*.

[8] Chau, C.-K. A Study of Unintended Behaviour in Networking

[9] Chau, C.-K., Gibbens, R., and Griffin, T. G. (2006). Towards a unifying theory for policybased routing. In *Proc. IEEE INFOCOM*.

[10] Chi, Y., Oliveira, R., Zhang, L., Cyclops: The AS-Level Connectivity Observatory. *In ACM SIGCOMM Computer Communication Review, Volume 38 Issue 5, October 2008.*

[11] Cobb, J. A., Gouda, M. G., and Musunuri, R. (2003). A stabilizing solution to the stable path problem. In *Proc. Self-Stabilizing Systems*, volume 2704, Springer-Verlag.

[12] Cobb, J. A. and Musunuri, R. Enforcing Convergence in Inter-Domain Routing. *In Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*

[13] Ee, C. T., Ramachandran, V., Chun, B.-G., and Shenker, S. (2006). Resolving BGP disputes. Technical report, EECS Department, University of California, Berkeley.

[14] Ee, C.,T., Ramchandran, V., Lakshminarayanan, K., Shenker, S., Resolving Inter-Domain Policy Desputes. *In Proc. of the 2007 conference (SIGCOMM '07) on Applications, technologies, architectures, and protocols for computer communications*.

[15] F. Baker, "Requirements for IP Version 4 Routers" RFC 1812, June 1995.

[16] Feigenbaum, J., Papadimitriou, C., Sami, R., and Shenker, S. (2002). A BGP-based mechanism for lowest-cost routing. In *Proc. Annual Symposium on Principles of Dis-tributed Computing (PODC)*.

[17] Feigenbaum, J., Ramachandran, V., and Schapira, M. (2006). Incentive-compatible interdomain routing. In *Proc. ACM Conference on Electronic Commerce*, pages 130–139.

[18] Feldmann, A., Maennel, O., Mao, Z. M., Berger, A., and Maggs, B. Locating Internet routing instabilities. In *Proc.ACM SIGCOMM* (August 2004).

[19] Griffin, T. and Huston, G, BGP Wedgies, RFC 4264, November 2005

[20] Griffin, T. G. and Wilfong, G. (2000). A safe path vector protocol. In *Proc. IEEE INFO-COM*.

[21] Griffin, T. G., Shepherd, F. B., and Wilfong, G. (2002). The stable paths problem and inter-domain routing. *IEEE/ACMTrans. Networking*, 10(2):232–243.

[22] Griffin, T., G., Wilfong, G., Analysis of the MED Oscillation Problem in BGP. ICNP 02, 2002.

[23] Jian Wu, Mao, Z.,M., Rexford j., Jia Wang, Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network. *In NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*

[24] Lupu, E. and Sloman, M. (1999). Conflicts in policy-based distributed systems management. *IEEE Trans. Software Engineering*, 25(6):852–869.

[25] McPherson, D., Gill, V., Walton, D., and Retana, A. (2002). RFC 3345: Border Gateway Protocol (BGP) persistent route oscillation condition.

[26] McPherson, D., Gill, V., Walton, D., Retana, A., Border Gateway Protocol (BGP) Persistent Route Oscillation Condition, RFC 3345, August 2002

[27] Rawat, A. and Shayman, M., A., Preventing Persistent Oscillations and Loops in BGP Configuration with Route Reflection. *Department of Electrical and Computer Engineering, University of Maryland, A.V. Williams Building, College Park, MD 20742, United States*

[28] Rekhter, Y. and Li, T. (1995). RFC 1771: A Border Gateway Protocol 4 (BGP-4).

[29] Ripe NCC. http://www.ripe.net/ripencc/pub-services/np/ris/.

[30] Siganos, G. and Faloutsos, M., Analyzing BGP Policies: Methodology and Tool. *In INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*

[31] University of Oregon Route Views Archive Project. www.routeviews.org.

[32] Uttaro, J., Schrieck, V., Francois, P., Fragassi, R., Simpson, A. Mohapatra, P., Best Practices for Advertisement of Multiple Paths in BGP, Draft-uttaro-idr-add-paths-guidelines-02, January 2011

[33] Walton, D., Chen, E., Retana, A., Scudder, J., Advertisement of Multiple Paths in BGP, draft-ietf-idr-add-paths-06, September 2011

[34] Walton, D., Chen, E., Retana, A., Scudder, J., BGP Persistent Route Oscillation Solution. Dtaft-walton-bgp-route-oscillation-stop-05, December 2011

[35] www.bgpvista.com/simbgp.php

[36] Yan, H., Oliveira, R., Burnett, K., BGPmon: A Real-time, Scalable, Extensible Monitoring System. *In Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications & Technology*.

[37] Yilmaz, S. and Matta, I., An Adaptive Management Approach to Resolving Policy Conflicts. *Technical Report BUCS-TR 2006-008.*