Combining SPF and Source Routing for an Efficient Probing Solution in IPv6 Topology Discovery

by

M. F. Rabbi Ur Rashid

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Carleton University

Ottawa, Ontario

The undersigned recommend to
the Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Combining SPF and Source Routing for an Efficient Probing Solution in IPv6 Topology Discovery

submitted by

M. F. Rabbi Ur Rashid

in partial fulfillment of the requirements for the

degree of Master of Applied Science

____

Chair, Dr. Roshdy Hafez, Department of Systems and Computer Engineering

____

Thesis Supervisor, Dr. Chung-Horng Lung

____

Thesis Supervisor, Dr. Marc St-Hilaire

Carleton University
May 2014

Abstract

For efficient network management, knowing the full topology of the network is important. Topology discovery using *source routing* and routing protocols are two well known methods to discover layer 3 connectivity. *Source routing* has the probing space explosion phenomenon that generates a large volume of traffic. As a result, *source routing* based approach takes a significant amount of time for network operators to discover and troubleshoot the whole network. Although routing protocol based approach like OSPFv3 discovers the network connectivity, the full IPv6 address cannot be discovered, as the approach only discovers the prefix portion of IPv6 addresses. This thesis proposes an efficient probing space reduction algorithm by combining *source routing* and OSPFv3. The idea is to apply *source routing* based on the information obtained from OSPFv3 based discovery for IPv6. Experimental results show that the proposed algorithm reduces redundant probing significantly which is useful for network management.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors Prof. Chung-Horng Lung and Prof. Marc St-Hilaire. Without their constant support and motivation it was never possible for me to complete this thesis. I would like to thank them for their continuous guideline and feedback. Without their detailed involvement in this research, I would be lost long ago.

I would like to thank Dr. Biswajit Nandy and Nabil Seddigh from *Solana Networks* for their immense support and help. Their guideline was really important to finish this thesis and solve critical problems.

I would also like to thank Md. Arifur Rahman Nafish and Muhammad Raisul Alam for valuable discussions and insightful thoughts from the very beginning of this research. I would also like to thank Ravikiran Adhyapak for being extremely supportive throughout this research.

Finally, I would like to thank my parents and brother for having belief in me throughout the time. I am grateful for their enormous support and encouragement.

Table of Contents

Abstract iii

Acknowledgements iv

Table of Contents v

List of Tables viii

List of Figures ix

List of Acronyms xi

1 Chapter: Introduction 1

1.1 Motivation 2

1.2 Research Objectives 4

1.3 Contribution 4

1.4 Thesis Overview 5

2 Chapter: Literature Review on Topology Discovery and Source Routing 6

2.1 IPv4 Network Topology Discovery 7

2.1.1 SNMP-MIB Based Approach 7

2.1.2 ICMP and Traceroute Based Approach 11

2.1.3 Routing Protocol Based Approach 12

2.2 IPv6 Network Topology Discovery 12

2.2.1 Hierarchical Approach 13

2.2.2 Traceroute and Source routing Based Approach 16

2.2.2.1 Probing from Multiple Sources 19

2.2.3 Routing Protocols Based Approach 20

2.2.4 ICMPv6 Based Approach 21

2.3 Source Routingin IPv6 23

2.4 Probing Space Explosion of Source routing 25

2.5 Source routing Mechanism for Network Management and Troubleshooting 27

2.6 Summary 29

3 Chapter: A Source Routing Based Algorithm for Efficient Probing Space for IPv6 31

3.1 Overview 31

3.2 Background of Probing Space Reduction 32

3.3 Advantage of Probing Space Reduction in Terms of Network Troubleshooting 35

3.4 Open Shortest Path First for IPv6 36

3.5 Architecture of OSPFv3 based Topology Discovery and Source Routing based Algorithm 41

3.6 Topology Discovery Algorithm to Reduce Probing Space of Source Routing 43

3.7 Explanation of the Algorithm 46

3.7.1 OSPFv3 Based Algorithm 46

3.7.2 Source Routing Based Algorithm 47

3.8 Design and Setup of the Proposed Algorithm 52

3.8.1 OSPFv3 Based Topology Discovery 53

3.8.1.1 Methodology of OSPF Based Topology Discovery 54

3.8.2 Minimizing Redundant Probing of Source Routing 57

3.9 Summary 59

4 Chapter: Performance Analysis of Topology Discovery Algorithm and Probing Space Reduction Algorithm 60

4.1 Analysis of OSPFv3 Based Topology Discovery Algorithm 60

4.2 Analysis of Probing Space Reduction Algorithm 64

4.2.1 Probing Space Versus Number of Initial Seeds 65

4.2.2 Probing Space Versus Location of Initial Seeds 71

4.2.3 Probing Space by Traces Versus by the Traditional Approach 73

4.2.4 Probing Space in Terms of Links Not Probed by Traces 75

4.2.5 Performance Analysis with Manhattan Grid Topology 78

4.3 Performance Analysis in Terms of Related Works 79

4.4 Summary 81

5 Chapter: Conclusion and Future Work 85

5.1 Limitations 84

5.2 Future Work 85

References 87

List of Tables

Table 3.1	OSPFv3 LSA Types..	39
Table 3.2	Connectivity Information of a Network Running OSPFv3 ……….	57
Table 4.1	Characteristics of Test Networks…………………………………..	61
Table 4.2	Comparison of Probing Space Reduction Algorithms....................	80

List of Figures

Figure 2.1	Network graph for a typical administrative domain..........................	8
Figure 2.2	Examples of valid and invalid connections between two bridges with different forwarding entries………………………...................	10
Figure 2.3	Minimum amount of information of three switches for connectivity determination…...	11
Figure 2.4	The LORIA IPv6 test platform with one LA in each interface of a router…..	14
Figure 2.5	Incomplete Discovery Because of Equal Weighted Path……...........	17
Figure 2.6	Increased coverage by Source routing………………....................	24
Figure 2.7	Probing Space Reduction in the Same Basic Path………………....	26
Figure 2.8	Troubleshooting with Source Routing…………………..................	28
Figure 3.1	Discovering IPv6 address by multiple probing sources....................	34
Figure 3.2	Incomplete IPv6 Address Discovery by OSPFv3 based Algorithm..	37
Figure 3.3	Architecture of Topology Discovery and Probing Space Reduction.	41
Figure 3.4	Flowchart of OSPFv3 Based Topology Discovery Algorithm……..	46
Figure 3.5	Connectivity Information from OSPFv3…………………….........	47
Figure 3.6	Flowchart of Source Routing Based Algorithm……….…………....	48
Figure 3.7	Reduction of Probing Space..………..	49
Figure 3.8	Discovering Topology of an OSPF Area…………………………...	54
Figure 3.9	Discovered Connectivity Information of an OSPFv3 Area………....	56
Figure 3.10	Implementation Procedure of Source Routing Based Algorithm..….	58
Figure 4.1	Network Topology Discovered by OSPFv3 Based Algorithm……..	63
Figure 4.2	Example of Seed Selection Method with 8 Initial Seeds......…….....	65

Figure 4.3 Probing Space vs. Initial Seeds for a Network of 70 Routers……… 66

Figure 4.4 Average Probing Space for Different Network Size…………….…... 68

Figure 4.5 Confidence Interval of Probing Space for Networks of 50 Routers.. 70

Figure 4.6 Confidence Interval of Probing Space for Networks of 60 Routers.. 70

Figure 4.7 Confidence Interval of Probing Space for Networks of 70 Routers... 71

Figure 4.8 Influence of Seed Location on Probing Space.................................. 72

Figure 4.9 Average Probing Space vs. Seed Location....................................... 73

Figure 4.10 Average Probing Space by Traces vs. by Traditional Approach for Networks of 50 Routers……………………….......................... 74

Figure 4.11 Average Probing Space by Traces vs. by Traditional Approach for Networks of 60 Routers….. 74

Figure 4.12 Average Probing Space by Traces vs. by Traditional Approach for Networks of 70 Routers……………………….............................. 75

Figure 4.13 Average Number of Links not Probed by Traces for Networks of 50 Routers.. 76

Figure 4.14 Average Number of Links not Probed by Traces for Networks of 60 Routers.. 77

Figure 4.15 Average Number of Links not Probed by Traces for Networks of 70 Routers.. 77

Figure 4.16 Probing Space for a 6 x 6 Manhattan Topology….............................. 78

Figure 4.17 Number of Links not Probed by Traces for a 6 x 6 Manhattan Topology……... 79

List of Acronyms

AS	Autonomous System
ARP	Address Resolution Protocol
AFT	Address Forwarding Table
BDR	Backup Designated Router
BGP	Border Gateway Protocol
CM	Central Manager
CPU	Central Processing Unit
DOS	Denial of Service
DR	Designated Router
GUI	Graphical User Interface
HL	Header Length
IGP	Interior Gateway Protocol
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Provider
IS-IS	Intermediate System to Intermediate System
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6
LAN	Local Area Network
LA	Local Agent

LSA	Link State Advertisement
LSDB	Link State Database
MTU	Maximum Transmission Unit
MIB	Management Information Base
NDP	Neighbor Discovery Protocol
OSPF	Open Shortest Path First
OSPFv2	Open Shortest Path First for IPv4
OSPFv3	Open Shortest Path First for IPv6
QoS	Quality of Service
RIP	Routing Information Protocol
RSO	Routing flag, Solicited flag, and Override flag
SNMP	Simple Network Management Protocol
SPF	Shortest Path First
TTL	Time To Live
UDP	User Datagram Protocol
WAN	Wide Area Network

Chapter 1

Introduction

Internet Protocol version 6 (IPv6) is the successor of IPv4. IPv6 has gradually been replacing its predecessor with its new and advanced features for the current Internet world. Because of the large address space of IPv6 combined with a different header structure and a new addressing architecture, IPv6 introduces a set of new protocols as well as improvement and modifications of some existing protocols. As a result, the adaptation of IPv6 has opened several research fields such as topology discovery, network management, security, quality of service, and so on.

For an efficient network management, a complete view of the network is essential. *Source routing* is an important mechanism for both topology discovery and network management [3]-[5]. It is a widely adopted approach in both IPv4 and IPv6. *Source routing* is an improved *traceroute* mechanism, where a sender can specify the path for an Internet Control Message Protocol (ICMP) packet, rather than depending on dynamic routing. It is also useful in network management to troubleshoot networks to find failure points, calculate latency, locate routing loops, etc.

Another way of discovering the topology of the network is to rely on routing protocols. This is a passive discovery approach, where a discovery algorithm only listens to all the packets that traverse through the network to build the topology without generating any additional packet in the network [4][24][25].

1.1 Motivation

An efficient management of IP networks involves knowing the full topology of the network and constantly monitoring the network to detect possible problems and troubleshoot them. The management mechanism makes topology discovery an important research field for both IPv4 and IPv6 networks.

OSPFv3 (Open Shortest Path First for IPv6) based topology discovery is a passive discovery approach that reveals the connectivity of OSPF (Open Shortest Path First) networks. Unlike *source routing*, OSPFv3 does not send any additional traffic to the network to discover the network topology. The OSPF based discovery approach identifies the full router level connectivity without being intrusive. However, OSPF based discovery methods cannot discover the full IPv6 address of the router interfaces, as they only provide the prefix portion of the IPv6 address for the links. Each link in the network has an IP address which consists of a prefix portion and the host portion.

Although in IPv6, *source routing* cannot discover all the addresses of the network, it is still needed to discover the full IPv6 addresses of the router interfaces. However, *source routing* often proves to be inefficient because of its probing space explosion problem [3][5]. *Source routing* needs to generate a large number of probing packets to probe the nodes and links of a network.

In *source routing*, the sender can specify the path to reach the destination by inserting IP addresses of the intermediate nodes into the routing header. A series of ICMP packets are sent with increased hop limit. A node sends back an ICMP *Time Exceeded* (*Hop limit exceeded in transit*) message to the source, if the hop limit in the generated packet is reached [26]. Thus, every intermediate node will send back an ICMP *Time Exceeded* message to the source, from which the source gets to know about the address of the originating interface of that node. While traversing through the mentioned nodes in the routing header, a *source routing* packet might traverse through additional nodes that are not in the list. All these nodes will send back the *Time Exceeded* ICMP packet to the sender. This way, a single probe of *source routing* discovers additional nodes in the network. A pair of addresses in *source routing* with one as the intermediate node and the other one as the destination node is referred to as a probing pair [5]. While traversing through these two nodes, a *source routing* packets might send back the information of additional nodes that it had to go through. A network management or discovery tool will generate more probing pairs based on the information of the discovered nodes to discover more nodes in the network. To probe the nodes and links in a network, the number of probing pairs a network management tool needs to generate is called "Probing Space" [5]. Probing space expands whenever a new node is discovered. This in turn generates a lot of probing pairs to probe all the links in the network. Therefore, it creates a high volume of traffic in the network and increases the discovery time significantly. Several research works have been conducted to reduce the number of redundant probing path of *source routing* [3][5].

Substantial amount of research works on topology discovery for both IPv4 and IPv6 networks has been presented [1]-[25]. It was observed that the *source routing* based approach and the routing protocol based approach are the two widely deployed layer 3 approaches among others for network discovery. It was also observed that, apart from being an important discovery approach, *source routing* is also used as an important network troubleshooting and performance analysis mechanism to measure delay, latency, packet loss and so on [37]-[42].

However, the probing space explosion of *source routing* makes it highly inefficient in terms of network management. For instance, ATLAS [3] took eight weeks to discover a network topology with 2,420 routers. ATLAS had to probe a total number of 308,887 paths to discover the topology, which creates a huge amount of traffic and takes a lot of time to discover the topology. Moreover, the explosion phenomenon of *source routing* creates high computational complexity which results in limited practical applications for network management and troubleshooting.

Hence, the motivation of this thesis is to find an efficient topology discovery algorithm for IPv6 networks. The objective of the algorithm is to reduce the probing space explosion of *source routing* for a network topology in such a way that it discovers the most number of IPv6 addresses that could not be discovered by the OSPFv3 based discovery approach. Moreover, the algorithm should also perform as an efficient tool for troubleshooting and performance analysis.

1.2 Research Objectives

To overcome the potential probing space explosion problem for IPv6 networks, the main objective of this research is to build an efficient network topology discovery algorithm for IPv6 networks. The algorithm traverses the complete router level topology of an IPv6 network by going through all the router interfaces. The word "efficient" means that we want to learn about the topology information as much as possible, while reducing the amount of probing required.

The proposed novel probing space reduction algorithm is a combination of both the OSPF based discovery algorithm and *source routing*. The proposed algorithm makes use of Dijkstra's Shortest Path First (SPF) algorithm to obtain the router connectivity information and then reduce redundant probing using *source routing*. To the best of our knowledge, no such hybrid technique has been reported in existing literature.

1.3 Contributions

The contribution of this thesis lies in the development of a probing space reduction algorithm of *source routing* that can be used as an efficient topology discovery and network management tool. The algorithm has significantly reduced the probing space. Specific contributions of the thesis include:

- Development of an emulation environment using Quagga [32] to obtain the real SPF information for topology discovery of an IPv6 network.
- Development of a novel hybrid technique: a technique that combines both the SPF information and *source routing* to reduce the probing space for IPv6 networks.

1.4 Thesis Overview

The remainder of this thesis is organized as follows. In Chapter 2, a literature review on topology discovery for both IPv4 and IPv6 networks and the probing space explosion problem of *source routing* have been discussed. This chapter also focuses on how *source routing* mechanism is used in network management and troubleshooting. Chapter 3 presents the proposed algorithm on probing space reduction of *source routing* as well as a router level topology discovery algorithm for networks running OSPFv3 protocol. Chapter 3 also explains the design of both algorithms in details. Results and performance analysis are presented in Chapter 4. Finally, Chapter 5 concludes this thesis with possible future works.

Chapter 2

Literature Review on Topology Discovery and Source Routing

For an efficient network management and performance analysis, an accurate topology of the network is a basic requirement. Without a perfect picture of the network topology, it is not possible for the network operators to pin point the exact location of a problem and predict possible problems before they occur. Topology knowledge is also important for analyzing application performance and selecting the best path for a particular application. As networks are becoming more and more complex now-a-days, having the true topology of the network is getting more important for configurations and maintaining security of the network. *Source routing* is one of the important topology discovery tools in IPv6, which has a probing space explosion problem that generates a great deal of traffic in the network. In this review, the focus will be to go through the existing approaches on topology discovery for IPv4 and IPv6 networks and the probing space explosion problem of *source routing*. Moreover, the importance of *source routing* in terms of network management and troubleshooting is also elaborately discussed. This review will discuss both the Local Area Network (LAN) and Autonomous System (AS) level discovery mechanisms. To get a complete picture, this chapter will start by some topology discovery algorithms for IPv4 networks to differ between the approaches of IPv4 and IPv6 topology discovery.

Topology discovery can be divided into two categories: i) Layer 3 topology discovery which is the IP level discovery of the routers, and ii) Layer 2 topology discovery which includes the discovery of switches, hubs, hosts and also the connection between them and their link layer addresses. The main problem of IPv6 topology discovery is its huge address space. Because IPv6 addresses have a 64-bit host portion, it is impossible to search for all the addresses in a given network. Moreover, a different header structure and the difference in other protocols make the IPv6 discovery approach significantly different than IPv4. For example, unlike IPv4, IPv6 has a fixed Header Length (HL) of 40 bytes [28]. IPv4 headers vary from 20 bytes to 60 bytes according to the size of the options inside the header. Moreover, IPv6 uses Neighbor Discovery Protocol (NDP) instead of Address Resolution Protocol (ARP) for layer 2 address resolution.

Although this research focuses on layer 3 IPv6 networks, this chapter presents both layer 2 and layer 3 discovery techniques for IPv4 and IPv6 networks along with the probing space explosion phenomena of *source routing*.

In the rest of this chapter is organized as follows. The first two sections introduce related work on IPv4 and IPV6 topology discovery respectively. Then, a detailed discussion on *source routing*, probing space explosion of *source routing* and its use for network management and troubleshooting is presented.

2.1 IPv4 Network Topology Discovery

2.1.1 SNMP-MIB Based Approach

Several works have been proposed for IPv4 layer 2 and layer 3 network topology discovery. Accessing the bridge information through SNMP-MIB (Simple Network Management Protocol - Management Information Base) and deriving a topology of the network in a geometric approach with the information collected through SNMP is one of the techniques that was proposed in [8] and [9].

The authors in [8] describe the topology discovery for both layer 2 and layer 3 networks. It follows a geometric approach and makes use of the SNMP-MIB. The paper uses the idea in terms of having multiple subnets in a single switch domain in a heterogeneous environment. Figure 2.1 shows the existence of multiple subnets in a single switch domain. As can be seen, a subnet can be composed of one or multiple switches that are directly connected (subnets 2 and 3) and by switches that are not directly connected (subnet 1).

The problem of having multiple subnets from the topology discovery perspective is that switches across the subnet do not keep information of other switches in a different subnet. Hence, accessing through SNMP will not provide the complete information. For example, in Figure 2.1,

S1 and *S2* do not keep information about the MAC address of each other even if they are directly connected, as they communicate through one or more routers (*R1* and *R2*). Both[8] and [9] adopt a geometric-based approach to solve such situations.

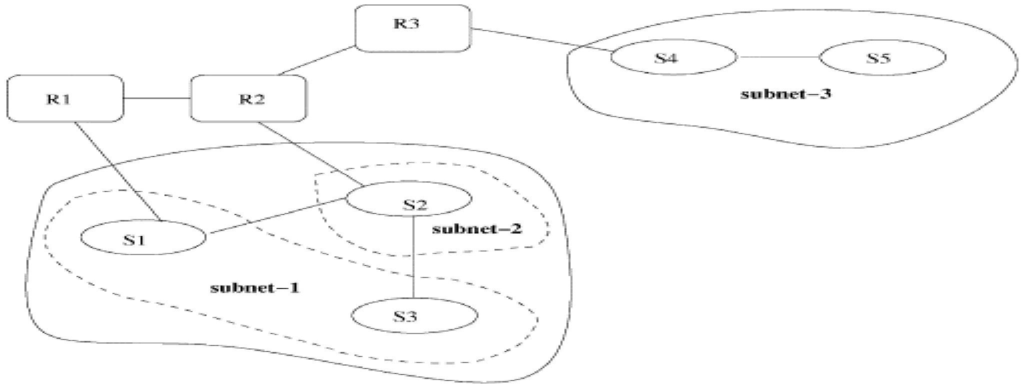


Figure 2.1 Network graph for a typical administrative domain [8]

On the other hand, for a typical subnet like the one that consists of *S4*and *S5* in Figure 2.1, the paper proposes a discovery algorithm that provides both the router level and the switch level discovery. To discover the set of routers, the algorithm assumes that it knows the IP address of at least one router. The key steps of the algorithm are outlined as follows:

- From the *ipRouteTable* entry in MIB-II of the router, the algorithm knows about the neighboring routers. It repeatedly finds the neighboring routers this way and at the end, a router finds all the routers in an autonomous system.
- Next, to identify the switches, the algorithm finds all the interfaces of the routers through the *ipAddrTable* entry in MIB-II. Then based on the subnet masks and IP address formats, it calculates all the IP addresses under those interfaces.

- After this, by checking at the Bridge MIB, the algorithm determines which of these IP addresses are switches.
- After that, the system determines the interconnections between the switches and checks the Address Forwarding Tables (AFTs) of the switches.

The authors in [8] state that, if the completeness requirement is achieved, then it is possible to determine the interconnection between the bridges based on the "Direct Connection Theorem". To describe it, let X and Y be the ports of two different bridges. X and Y have the forwarding set of addresses F_x and F_y and they are complete. This means that all the bridge information that are reached via port X and Y is in F_x and F_y consecutively. Now, if $F_x \cap F_y = \Phi$ and $F_x \cup F_y = N$, then these two bridges are directly connected, where N is the total number of bridges that these two ports are connect to. In this way, the method presented in [8] accesses the bridge information, determines their interconnection and builds a complete topology. To determine this interconnection, all the switches must have complete information about the other switches. But to determine the interconnections between the routers and the switches, it takes a different approach. In other words, if an interface of a switch is a leaf interface (i.e., an interface that does not connect to another switch) and has the MAC address of a router in its forwarding set database, then that interface and the router are connected. If an interface of a switch is not connected to another switch, it means it is either connected to a host or a router. Further, if the switch contains the MAC of a router, then it is connected to that router.

One of the limitations of the approach proposed in [8] is that it discovers only the router and switch level topology. The approach does not discover any information about the hosts that the switches are connected to. Another limitation is that, to determine the interconnection between the switches, the address forwarding tables of the switches have to be complete which is not always the case.

The authors in [9] propose another geometric solution which overcomes the limitations of the completeness requirement that was proposed in [8]. The completeness requirement of the bridges is to discover the layer 2 topology, which requires each bridge to have entries about all other bridges. This means that each bridge will have the information about all the other bridges in their address forwarding tables. However, the problem in the completeness requirement is that it is highly unlikely that the bridges will have the complete information. As the bridges do not maintain information permanently in their AFTs, constant traffic needs to be generated. As the network grows larger, it gets even more challenging. The authors in [9] propose an algorithm which states that a bridge needs to know the information of only three bridges to show how they are connected.

The following figure (each bridge having information about only two bridges) demonstrates how their algorithm works.

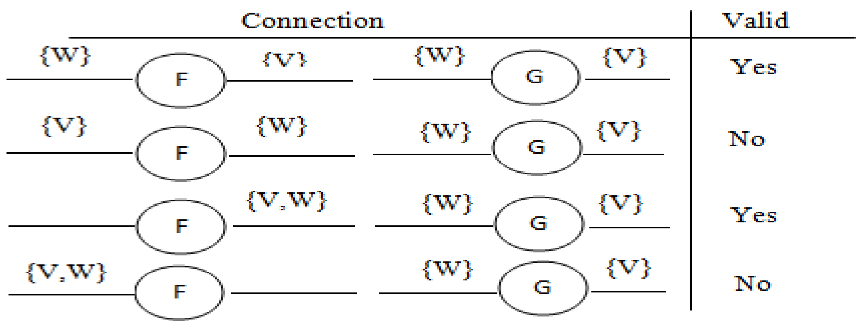


Figure 2.2 Examples of valid and invalid connections between two bridges with different forwarding entries [9]

In Figure 2.2, there are four examples. The connection of the fourth example in the figure is not possible as switch *V* cannot be in two different directions in the network. Here, two switches forward the same address *V* in opposite directions. So if two ports of two switches claim that the same machine is in two different places in the network, we can say that these two switches cannot be connected. The second example is not valid for the same reason. The first example in Figure 2.2 has no contradiction. However, it is possible that there are other switches between them. The third example shows that node *W* is in between them. The method in [9] is based on this idea and proved that a minimum information of three switches (Figure 2.3) in the AFT is necessary to determine how these switches are connected. In Figure 2.3, *A* and *B* cannot be connected as they have the same address *Y* and *Z* in opposite direction. However, it should be noted that their approach can have other switches in between. The paper also shows an approach to determine if two ports are directly connected or if they have other switches in between.

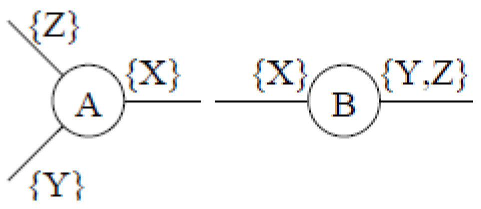


Figure 2.3 Minimum amount of information of three switches for connectivity determination [9]

The paper assumes that most bridges in an Ethernet LAN will have this minimum information (i.e. three switches) in their AFT. If some of the bridges don't contain this minimum information, the topology can be falsely identified. Also, the paper solely depends on SNMP which can create problems if there is any switches that do not authorize access to its SNMP-MIB. The author of the paper also appreciated the fact that the paper needs some improvement to work in the VLAN environment properly.

2.1.2 ICMP and Traceroute Based Approach

So far, only two papers on the layer 2 and layer 3 discovery in IPv4 networks have been discussed. However, a lot of work has been proposed in the related area. Burch and Cheswick proposed a layer 3 discovery based on ICMP in [11].

One of the main layer 3 discovery tools in IPv4 is *traceroute*. The concept of *traceroute* based discovery was first proposed by Jack Rickard in [18].The Mercator project uses *traceroute* to produce an Internet map [10].The paper in[19] proposed a *traceroute* based algorithm that discovers the topology in a network-friendly manner. As *traceroute* generates packets in the network, it might be a problem for large scale implementation. The paper focused on maximizing the number of *traceroute* monitors on the routers of the network. The paper in [20] also tried to improve the probing efficiency of *traceroute* in topology discovery. The paper in [21] introduced a new method called Backtrace Tree increasing the probing efficiency of *traceroute*.

CAIDA developed Skitter [12] which is a distributed system for network management. The authors in [13] proposed an algorithm called Octopus which combined SNMP, *traceroute*, measurement and heuristics to determine the topology. The paper in [15], discovered the topology by path probing. Moreover, the papers in [22] and [23] also used *traceroute* as part of their discovery algorithm.

2.1.3 Routing Protocol Based Approach

Topology Discovery using routing protocols is already a well adopted layer 3 discovery method for IPv4 networks. RIP-based discovery algorithms are used when the network shares the route information using RIP (Routing Information Protocol) protocol. RIP is a distance-vector routing protocol. As it uses hop count as its routing metric, the connection between the routers can be deduced through the hop counts. The RIP protocol is available for small to medium scale networks while the OSPF protocol is mostly used in large enterprise networks [24]. OSPF was deployed because of the well known limitations of RIP such as small network size, long convergence time, etc. OSPF is the most widely used Interior Gateway Protocol (IGP) in large networks. Several authors have discussed about OSPF based topology discovery in IPv4 networks [25]. Both RIP and OSPF are passive discovery methods as they build the topology by collecting RIP or OSPF messages.

In the next section, IPv6 topology discovery approaches will be discussed. The discovery process in IPv6 is different than IPv4 due to its large address space. Approaches based on hierarchical structure, *source routing* and routing protocols are some of the effective approaches proposed for IPv6 topology discovery.

2.2 IPv6 Network Topology Discovery

Topology discovery in IPv6 networks is still a new field and a great deal of research work is still going on. Many discovery methods for IPv4 cannot be directly implemented in IPv6 networks. For example, ICMP has been the most widely deployed discovery tool for IPv4 networks. However, it cannot be used in IPv6due to the large address space. ICMP packets for IPv4 are broadcasted throughout the network and the probing machine waits for the replies from the nodes to construct the network map. In IPv6, the 64-bit host portion (out of the 128 bit address) would generate 2^{64} ICMPv6 echo requests which is tremendously expensive in both network resource consumption and time [1]. Another example of IPv6 discovery tool that cannot be implemented in IPv6 is the Rocketfuel [17] probing tool. Rocketfuel used more than 480 *traceroute* servers in the world [23]. Rocketfuel uses the IPv4 header identifier field to help identify addresses belonging to the same router [3]. Unfortunately, the IPv6 header does not have any such field and therefore, this tool cannot be used in IPv6 discovery.

Different techniques have been developed in discovering the network topology for IPv6 networks. LORIA laboratory proposed an idea based on hierarchical structure [1]. Bell Lab also proposed a method based on *source routing* called Atlas [3]. There were other papers which adopted the ideas of hierarchical structure and *source routing* [2]-[5]. In addition, many papers also use different protocols like ICMP or MIB through SNMP [14] for the discovery. Finally, the layer 3 information can also be retrieved by different routing protocols, such as OSPF.

2.2.1 Hierarchical Approach

LORIA laboratory has proposed a topology discovery (both layer 2 and layer 3) based on hierarchical structure [1]. A two-level architecture is proposed with a Local Agent (LA) and a global agent, also called Central Manager (CM). There will be a LA in every link (subnet) and the CM will be somewhere in the network [1]. The idea is that the LAs will find all the information of their local link or subnet they are in and send it to the CM. The CM will process the information and build the complete topology. In Figure 2.4, the CM is implemented in the "Aria" router. Every subnet (local link) has one LA in it, which collects the subnet information individually and sends it to the CM. The CM will then build the complete topology.

In Figure 2.4, the CM is implemented in the Aria router, and the other routers like Asterix, Throgal, Treize and Garfield are implementing the LA. There is one LA in every subnet, as can be seen from the figure. These LAs send the information of their respective subnet to the Aria router which is the CM.

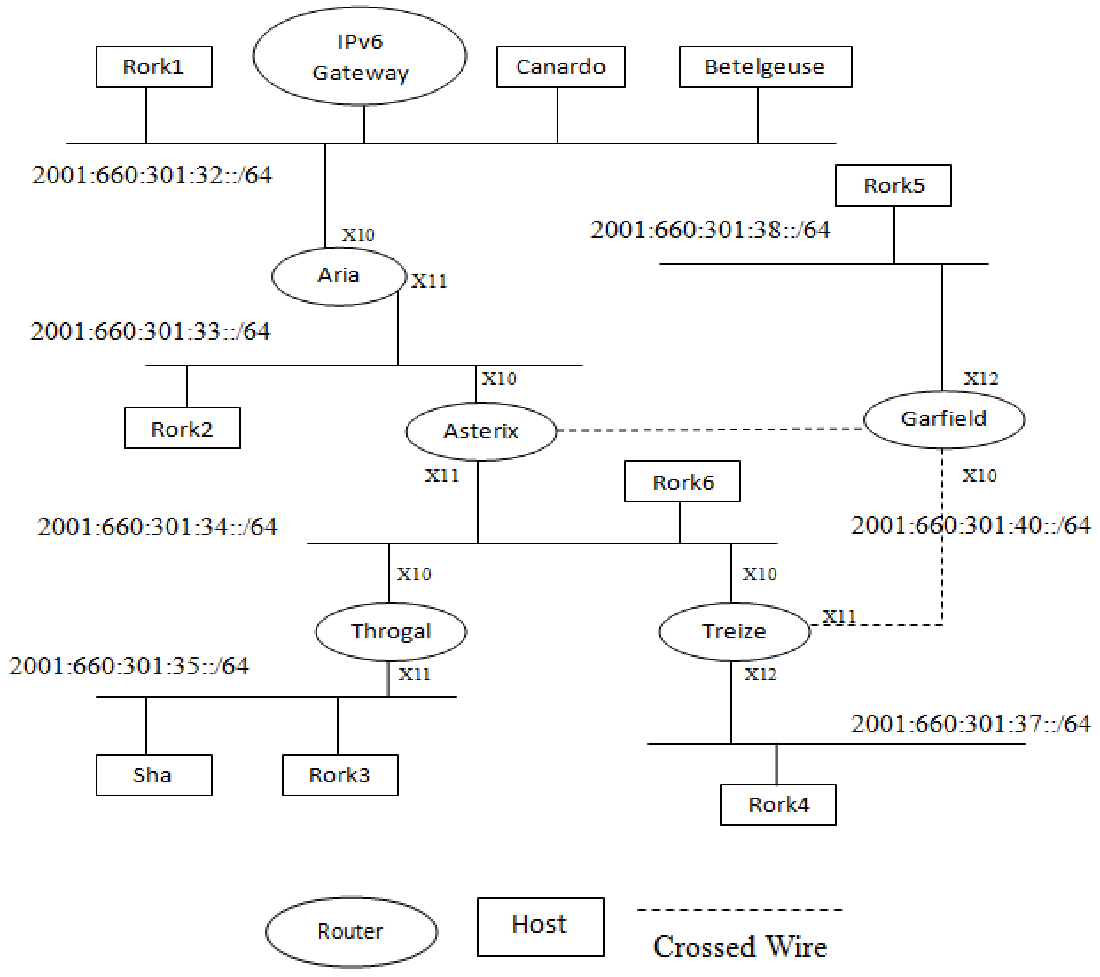


Figure 2.4 The LORIA IPv6 test platform with one LA in each interface of a router [1]

In the first phase of the algorithm, the LA sends ICMP echo request to all the nodes via the multicast address. To let the other nodes know about its presence, the LA will send a neighbor solicitation message. Upon receiving the different ICMP packets under the neighbor discovery protocol [7], the LA gets to learn about the nodes' link layer address, local link address, global IPv6 address and also MAC address. For example, the link layer address can be retrieved from the option field of the neighbor advertisement. Also, an unsolicited neighbor advertisement will give the IPv6 global address in its target address field. The LA will read the RSO (Router flag, Solicited flag, and Override flag) to determine if the sender is a router or a host. Up to this point, the algorithm is considered as an active algorithm, since it is sending requests to collect information. After this step, the algorithm proceeds to the second phase where the LA does not send any ICMP requests. This is the passive part of the algorithm. It means that the LA only listens to all the traffic (neighbor advertisement, router advertisement, etc.) on its interface and determines the information that was not retrieved in the active part. The LA also *traceroutes* the CM and learns the information about all the intermediate routers.

The main disadvantage of the paper is that it requires a probe seed in every subnet of the network, which makes the system complex and generates a lot of traffic. However, the load is shared among several workstations, which reduces the system runtime. Moreover, the algorithm did not define how it will work under the condition where there is a tunnel between IPv4 and IPv6 networks.

Another paper [2], which is also based on hierarchical structure, focused on both LAN and WAN (Wide Area Network) topology discovery. The local discovery module is pretty much the same as the previous one [1]. The main contribution of the paper is that it describes the topology in a WAN environment and also in the presence of IPv6 to IPv4 tunnels. Unlike the method described in [1], the CM is responsible to build the backbone of the network. The method examines the gateway's BGP (Border Gateway Protocol) route table and find out the possible IPv6 routers in the WAN. To build the topology of the WAN, the CM will *traceroute* all these routers and combining these with the information collected from the LAs. The LAs will also run the *traceroute* to the CM to find the intermediate routers.

The main difference between these two papers is that the authors in [2] also show a way to retrieve the information when there is an IPv6 to IPv4 tunnel. The algorithm retrieves the MAC address of the nodes from the local link address when the LAs run the local discovery module. Hence, the algorithm knows the IPv6 addresses and their corresponding MAC address, but it does not know their IPv4 addresses if those routers run the dual stack protocol. To know the corresponding IPv4 address of a particular IPv6 address, the algorithm looks at the server's ARP buffer where there is an IPv4 address to MAC address mapping. As the MAC address of a device will always be the same, the system can easily match the IPv6 and IPv4 addresses of a device.

2.2.2 Traceroute and Source Routing Based Approach

Using *traceroute* [10] to find the network topology is one of the basic ideas of the IPv6 topology discovery. Even though *traceroute* was used in the LORIA [1] paper discussed in Section 2.2.1, the algorithm in this section is run by a single probe seed rather than using one per subnet. *Source routing* is adopted widely to discover IPv6 networks. One of the basic differences between IPv4 and IPv6 is that in IPv4, only 8% of the routers are source route capable while in IPv6 most of them are capable of *source routing*.

The paper in [3] discovers the layer 3 topology based on both *traceroute* and *source routing*. They designed a system called ATLAS, which addresses some of the basic scenarios like limiting the bandwidth and forwarding costs of ICMP, prolonged router non responsiveness, anonymous interfaces, instable routing and address equivalence and tries to minimize the effects caused by these.

The seed selection of the ATLAS system was from the 6Bone registry [3]. As the IPv6 network is growing larger and seed selection cannot be obtained from the network registry, they suggested random seed selection from the network. To increase the probe engine performance, the ATLAS system employs caching. For each trace, the system caches the hop distance to the via-routers which reduces the time and traffic for the subsequent traces if the via router is used again. To limit the bandwidth and forwarding cost of ICMP, the system avoids sending too many probes in a relative locality. Prolonged router non-responsiveness is another phenomenon which happens in the networks. It happens when the router is an anonymous router (routers that are nonresponsive to ICMP) or doesn't respond to probes over a prolonged period of time. ATLAS use push through mechanism to deal with the situation. It waits for a certain amount of time to get the response. If it doesn't get a response by this time, it increases the hop limit and proceeds to the next router.

Anonymous interface [16] can also occur because of the scope based addressing in IPv6. If the interface has a link local address and not the global address, the interface will not return its own address as the source address in the ICMP response. It will return the destination address the *traceroute* was performed to. This way, the address of that interface cannot be retrieved through *traceroute*. As the interface information remains unknown, one interface comes up in several probing and can be considered as multiple routers which reduces the accuracy. ATLAS performs several geometric and prediction mechanisms to merge these anonymous interfaces as a single interface.

The instable routing occurs when multiple routes have equal weights. The fundamental assumption of *traceroute* based probing is that, packets between two routes will always take the same route. But common routing protocols like OSPF, IS-IS support equal weighted multiple paths for traffic engineering purpose. In tracerouting, the probe engine can miss a lot of routers for each *traceroute* function because of equal weighted paths. In Figure 2.5, router *C* may not be identified for a *source routing* because of the same weight between *A* to *B* and *A* to *C*. To deal with this situation, the system dispatches multiple probes in batches for each hop. However, this creates a lot of traffic which prolongs the discovery time.

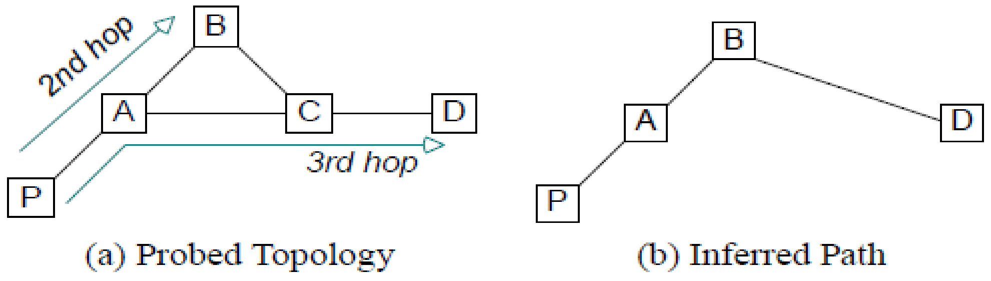


Figure 2.5 Incomplete Discovery Because of Equal Weighted Path [4]

One drawback of the paper is that it creates too many ICMP traffic to encounter some of the mentioned problems. Moreover, random seed selection can also lead to an incomplete discovery as appropriate seed selection plays a vital role in topology discovery. The main drawback of the ATLAS system is that it takes too much time to discover the entire topology. For the 6Bone network, it took 8 weeks to discover the entire map which was composed of 2,420 routers. This shows how time consuming the *source routing* method can be, if further action is not taken.

The paper in [5] addresses the major concern of *source routing* which is probing space. In *source routing*, any two routers are taken to source route and find more routers between them. It considers all the possible combinations of two routers. As the system finds more routers, more combinations of two routers are possible. The total number of these combinations is called probing space. The probe space grows rapidly for each successful probe. For example, assuming the number of nodes is n, the probing space is $n(n\text{-}1)$. If the system finds x additional routers, the probing space will be $(n+x)(n+x\text{-}1)$. This creates a lot of traffic and takes a lot of time which has been the major concern of *source routing* so far. The paper proposes some ideas to reduce the probing space.

The first idea is that no additional router can be found between two routers if both of them are in the same basic path. This means that if *source routing* is performed to a particular destination, the combinations of the intermediate routers from source to destination, cannot find any additional router.

The second idea is that if one router is not source route capable in a basic path, all the subsequent routers in that path are also considered to be unable to support *source routing*. This means that all the combinations that include these routers can be removed from the probing space.

This paper is useful to reduce the discovery time of IPv6 networks. However, the paper did not show their work in terms of anonymous routers. Anonymous routers do not respond to ICMP messages and are also not able to perform source route which leads to a lot of undetected routers.

2.2.2.1 Probing from Multiple Sources

The *traceroute* or *source routing* based discovery approach can be performed from a single source (probe engine) or multiple sources. In multi-source probing, more than one source will generate *traceroute* or *source routing* packets to discover the topology. Most existed distributed topology discovery applications deploy multiple probing hosts throughout the network. Paths are then probed from the individual points and the data later is unified into a single topology graph [49]. Probing all the nodes, links and interfaces of the nodes is important. Even if a probing engine probes all the nodes in a network, it still needs to probe the interfaces of the nodes to get information about the IP address for the interfaces and links.

The authors in [46] suggested that probing from multiple sources and amalgamating the results could improve the completeness of the map. The authors showed a comparison between probing from one node and probing from two nodes. They probed 256 IP addresses from two distributed nodes (connected to Rogers @Home and the Internet Gateway network in Vancouver). Probing from two nodes discovered 76% more nodes than probing from one node. Moreover, the number of links (connection between two nodes) discovered by two probing nodes was 90% more than that of one probing node. However, the efficiency decreases for multiple sources in terms of redundant discovery. 68% of the nodes were discovered redundantly 5 or more times when probing from one node. On the other hand, when probing from two nodes, this number increases to 83%. Reducing redundant probing is one of the focuses of this research.

Deployment of multiple probing sources is usually quite costly and complex [47]. The authors in [47] proposed a method to discover the maximum number of nodes while selecting the minimum number of probing sources. As the size of the network topology is often large compared to the number of probing sources, it is important to determine the placement and location of the probing sources to maximize the discovery. Authors in [48] proposed a method on how to place the probing sources to discover an accurate view of the topology.

2.2.3 Routing Protocol Based Approach

Although the routing protocol based approach is a popular and well adopted discovery approach in IPv4, not much work has been done for IPv6. The authors in [4] proposed a routing protocol based approach for next generation Internet. They discover the topology of a campus network by combining an improved *traceroute* based method with the OSPFv3 routing protocol. The main contribution of the paper was to solve the router alias problem and the crosslink problem that occurs in the classic *traceroute* based algorithm.

A router alias problem occurs when the system finds multiple IP addresses of different interfaces of a single router and draws the conclusion that multiple routers exist. In addition, the crosslink problem occurs when the *traceroute* function cannot identify some routers as they exist in between two basic path of *traceroute*.

In [4], *source routing* is adopted to solve the crosslink problem. As arbitrary nodes are selected as intermediate nodes for *source routing*, the crosslink problem is greatly reduced. In this paper, the *traceroute* based algorithm includes both *traceroute* and *source routing*. At first, all the border routers are *tracerouted* and each of the hop information is saved. The routers that are found are used as intermediate router for *source routing* which helps to discover a lot of new paths. The link information between all the routers can also be retrieved during these steps.

To solve the router alias problem (different IP addresses of a single router are considered as different routers), a single router is *tracerouted* via different paths and the router may respond with a different address as the source address of the reply message. So, it can be said that these addresses are originated from the same router. However, the algorithm relies a lot on the routing protocol part to solve the alias problem as *source routing* can't detect all the alias problems.

As the router alias problem is not completely solved in the *traceroute* based approach, the algorithm also uses information from the routing protocol (OSPFv3). Moreover, by combining *source routing* with a routing protocol, the algorithm gets more accurate topology of the network. At first, the machine that the algorithm is running into, sends the multicast address FF02::5for all the OSPF router in the area. Upon receiving an OSPF hello message, it sends a database description message to the end router and the link state request message following that. The topology is built based on the echo database description and the link state advertisement (LSA). The algorithm gets both the router information and the subnet information here.

Next, the router list achieved from the *traceroute* based algorithm and the router and subnet list from the OSPF based algorithm are combined. If the prefix of a router in the *traceroute* based method matches any subnet of the OSPF based method, then the algorithm finds the routers who belong to that subnet in the router list of the OSPF method. Then, the algorithm performs the hop count and compares the result between the *traceroute* and the OSPF based method. Most of the alias problems are solved here and the algorithm gets a complete picture of the network topology.

One advantage of the paper is that it doesn't need a probe seed in every subnet as needed in LORIA [1]. Moreover, the topology discovery based on *source routing* [2], does not always work as all the routers might not have *source routing* capability. To solve that issue, this paper combines the information from *source routing* with the information collected from OSPFv3. Using the routing protocol along with the *source routing* minimized the traditional router alias problem and the crosslink problem. This paper also tried to reduce the redundant probing of *source routing*.

2.2.4 ICMPv6 Based Approach

The paper in [6] describes a topology discovery mechanism based on ICMPv6. This paper also focuses on an efficient seed selection method to reduce redundant probing. It is based on layer 3 discovery and the algorithm finds all the routers in a reverse calculated way. It finds the farthest router at first and finds the previous routers by decreasing the hop limit by 1. It uses the information from theIPv6 address space of the "Whois". For probing, it chooses IPv6 addresses from the address space in a dimidiate method. For example, at first it probes i^{th} and $(i+n)/2$ th addresses and retrieves router information associated to those addresses. ICMPv6 says that if a device receives a UDP (User Datagram Protocol) packet with an unknown destination port number, the device returns a “port unreachable” error message and the header of the original message will be included in the response. At first, the device sends a UDP packet to a destination address (taken from “Whois”) with a destination port number 65,535 which is not normally used. So even if the destination address is alive, the corresponding router of the destination address will return the “port unreachable” error message. If the destination address is not alive, the router will send a “target unreachable” message. Both the target unreachable and port unreachable messages are ICMPv6 messages and they contain the header information of the original message. This header contains the information about the Header Length, which is the number of hops traversed by the UDP packet. The HL is similar to the TTL (Time to Live) in IPv4. So now the algorithm has information about the router and the number of hops to that router. The algorithm finds the previous router by decreasing the hop limit by one.

The main advantage of the paper is that it might have the ability to recover the significant delay of *source routing* mentioned in [3] where it took 8 weeks to discover the complete topology. A problem might be in the selection of the seed from “Whois”. The selection is solely based on dimidiate method so a lot of routers might remain undiscovered. The crosslink issue might be a problem in this case as the algorithm does not check all the possible options between the routers as the previous papers [3]-[5] did.

The paper also finds a mechanism to discover IPv4 to IPv6 tunnels. For this purpose, it uses the MTU (Maximum Transmission Unit) of the packet. ICMPv6 standard says that, if a router receives a packet bigger than it can handle (MTU), it responds back with a “Packet too big” error message. If an IPv6 packet is encapsulated into an IPv4 packet, the size of the packet is increased by 20 bytes (20 bytes is the minimum length of the IPv4 header). The algorithm detects this change and discovers that there is a IPv4-IPv6 tunnel.

2.3 Source Routing in IPv6

Source routing is an improved *traceroute* mechanism, where a sender can specify a path for an ICMP packet rather than depending on dynamic routing. A sender can specify a sequence of intermediate nodes to be traversed before reaching the destination.

Unlike IPv4, IPv6 handles the options in a separate header which are kept hidden from the original header. These headers are called extension headers and they are kept hidden in the payload. Thus, *source routing* in IPv6 is dealt in the extension header rather than the original header. The original header consists of another field called Next Header Field that tells the routers if there is any extension header in this packet. The router processes the extension header only if the router finds in the original header, that the packet has an extension header that is needed to be processed by the router [27]. There are six types of extension headers [28]:

- Hop-by-hop option header
- Routing header
- Fragment header
- Destination options header
- Authentication header
- Encrypted security payload header.

The routing header performs *source routing*. At first, the router knows through the next header field in the original header, that the packet is intended for *source routing* and needs to be examined by the router. Then the router looks at the extension header in the payload. The routing header contains all the addresses that the packet needs to go through, a routing type field and a segment left field. The router decreases the segment left field each time it forwards the packet to the next hop [28]. Each time the router changes the source and destination addresses of the routing header. The destination address is always the next hop address. The idea of extension header in IPv6 is developed for fast and efficient routing. The router will not process any option field, until and unless the next header field mentions this.

There are two types of *source routing*:

- Strict *source routing*: In strict *source routing*, the sender specifies the exact path the packet needs to follow. A list of addresses are inserted in the packet. The packet cannot go through any other node other than the specified addresses. If a router encounters an address in the list that isn't on a directly connected network, the router drops the packet and sends an ICMP source route failed message [37].
- Loose *source routing*: In loose *source routing*, the packet can traverse through other routers between any two addresses in the list. *Source routing* in IPv6 is a type of loose *source routing* [5].

The following figure depicts how *source routing* increases the coverage of topology.

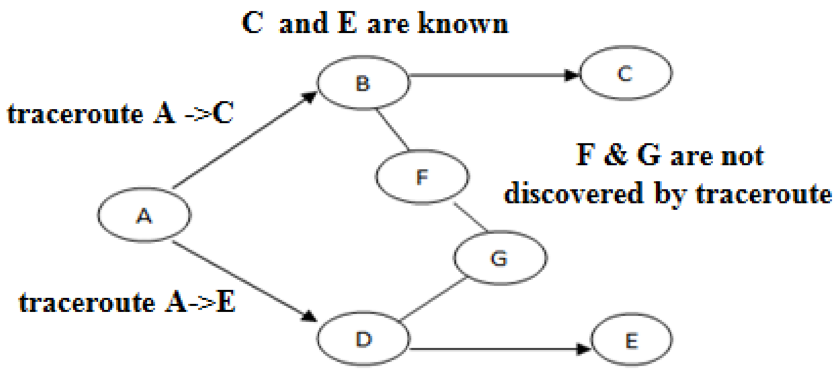


Figure 2.6 Increased Coverage by Source Routing

In the figure above, performing *traceroute* from *A* to *C* and *A* to *E* will discover only *B* and *D,* respectively. The nodes *F* and *G* cannot be discovered by the traditional *traceroute*. Whereas, in *source routing*, we can specify node *D* as an intermediate router and set node *B* as the destination router. Then the packet will traverse through nodes *F* and *G* which will give the information about these nodes. This way, the *source routing* mechanism uses the known nodes to generate probing pairs like *D-B* in this figure and sends *source routing* packets that traverse through those nodes.

2.4 Probing Space Explosion of Source Routing

As discussed in the previous section, *source routing* in IPv6 is a type of loose *source routing*. A list of two addresses can be inserted in the packet, where one is the address of an intermediate node and other one is the address of the destination node. A *source routing* packet will traverse through the intermediate node to the destination node. The pair of these two nodes is called probing pair [5]. Thus, if there are *n* nodes in a network, then the *source routing* mechanism will generate *n(n-1)* probing pairs. For a given number of nodes, the number of probing pairs that are generated is referred to as probing space. The probing space increases whenever a new node is discovered.

For a given network of n_0 nodes, the first round of *source routing* will generate $n_0\ (n_0-1)$ probing pairs. Now, after generating *source routing* packets to these nodes, if n_1 new nodes are discovered, then the probing space will increase to $(n_0+ n_1)(n_0+ n_1-1)$. After generating *source routing* packets to these nodes, if n_2 new nodes are found, then the probing space will be $(n_0+ n_1+ n_2)(n_0+ n_1+ n_2-1)$. Thus, each round of *source routing* is increasing the probing space rapidly. In general, if n_{i-1} new nodes are discovered in round *i*-1, then the size of the probing space in round *i* will be

$$C_i = \sum_{t=0}^{i-1}\left[n_t\left(\sum_{t=0}^{i-1} n_t - 1\right)\right], \mathrm{i} \geq 1 \qquad [5]$$

A concrete example is presented here to show how rapidly the probing space increases. If a network administrator knows about 20 nodes in the first round, then the probing space will be 20(20-1) = 380. If the first round discovers 10 new nodes then the number of probing pair will increase to (20+10)(20+10-1) = 870. This explosion of probing space is the main reason for Atlas [3] to discover their topology in eight weeks. Probing space explosion also generates a high amount of traffic in the network. The research proposed in this thesis focuses on the design of an efficient algorithm to reduce the probing space.

ATLAS [3] employs caching and parallelism to increase the performance of the probing engine. Probe engine is the system from where the *source routing* packets are generated.

To reduce the number of packets generated by *source routing*, ATLAS caches the hop distance to an intermediate router for each trace. If the same intermediate router is used for subsequent traces, then the cached hop limit will be the initial hop limit for these subsequent traces. Like *traceroute*, *source routing* is also performed with increased hop limit. The 1-byte Hop Limit (HL) field of IPv6 header is decreased by 1 each time a router forwards a packet to the next hop [27]. To reach a destination, *source routing* generates multiple packets increasing the hop limit each time. ATLAS caches this hop limit for an intermediate router. For another trace, if the same intermediate router is used, ATLAS starts the hop limit from the cached value rather starting the HL from the beginning. ATLAS also performs multiple traces concurrently to reduce the amount of time to probe the nodes.

As discussed in Section 2.2.2, the authors in [5] also proposed some ideas to reduce redundant probing. The first idea is that, no additional router can be found between two routers, if both of them are in the same basic path.

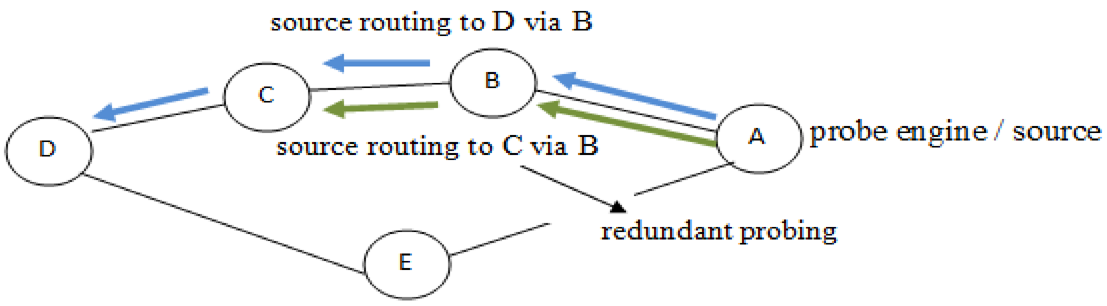


Figure 2.7 Probing Space Reduction in the Same Basic Path

Figure 2.7 shows node *A* as the probe engine or *source routing* packet generator. If a *source routing* packet is generated to the destination *D* keeping *B* as the intermediate node, then the subsequent traces are redundant. For example probing to *C* via *B* will not discover any new information about the intermediate nodes as the intermediate nodes are already known from the previous trace. Another idea proposed by the authors of this paper is that if *source routing* is disabled in one of the routers, then *source routing* for all the routers in a *source routing* path is redundant. For example, if *source routing* is disabled in node *B*, then probing to *D* and *C* via *B* will not be possible. This paper verified their algorithm with 1,000 nodes and 1,150 links. They pruned off a total of 410,376 probing pairs of *source routing* using their algorithm [5].

2.5 Source Routing Mechanism for Network Management and Troubleshooting

In the previous sections, we have seen how *source routing* is used to discover topology. However, *source routing* is also an important mechanism for network troubleshooting [37]-[42]. *Source routing* is an improved *traceroute* mechanism. *Traceroute* and ping are the two fundamental troubleshooting mechanism for network management [40].

The goal of *source routing* is to allow a network engineer to test a path through IP (Internet Protocol) routers to a remote destination [37]. Some ISPs (Internet Service Providers) like to have *source routing* available to troubleshoot problems in their own networks and neighbouring networks, especially when routing has broken inside one of those networks [41].

Source routing is similar to *traceroute*. A series of ICMP packets are sent with increased hop limit. A node sends back a ICMP *Time Exceeded Message* to the source, if the hop limit in the generated packet is reached [26]. If a node cannot forward a packet for some kind of network problem (link failure, node failure etc.), the node sends back a *Destination Unreachable* ICMP packet (type 1 ICMPv6) [29] to the source. The node might also send a *Time Exceeded Message* (type 3 ICMPv6) to the source if there is an alternative path to the destination. This way, a network operator gets to know about the location of the error.

Another way of troubleshooting is to route a test packet "out and back", through some set of gateways and back to the originating node. A series of such tests, tracing successive steps in the route that failed, should quickly locate the troublesome gateway [39].

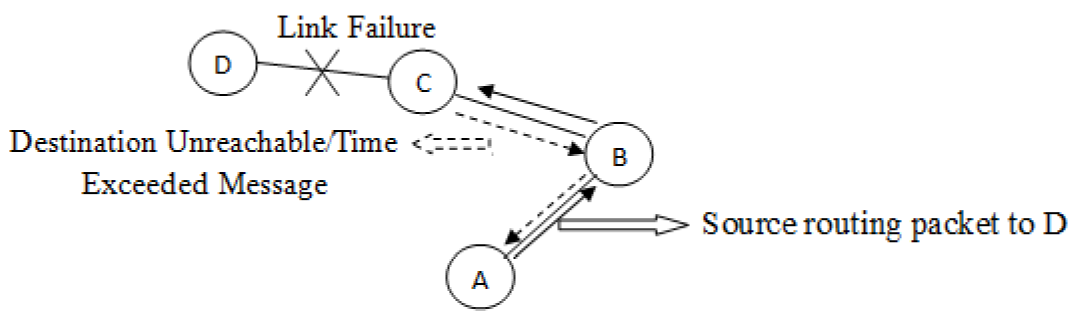

Link Failure
D
C
Destination Unreachable/Time Exceeded Message
B
Source routing packet to D
A

Figure 2.8 Troubleshooting with Source Routing

Figure 2.8 shows how a *Destination Unreachable* or *Time Exceeded Message* can be used for network troubleshooting. If node *A* sends a *source routing* packet to node *D*, node *C* will send a *Destination Unreachable* message, because the link between *C* and *D* fails. If there is another path from node *C* to *D*, then node *C* will send back a *Time Exceeded Message*.

ICMP messages like ping, *traceroute*, *source routing* are also used for latency calculation [42]. Quality of Service (QoS) is the transmission quality and service availability of a network. The transmission quality of a network is measured by packet loss, delay and delay variation [43]. *Traceroute* is a common ICMP based network management tool. *Traceroute* can measure the elapsed time between transmission and reception of an ICMP packet. Apart from discovering the path information, *traceroute* can also measure the delay and loss characteristics of a path [37]. *Source routing* is an improved *traceroute* mechanism that is used for better performance measurement and analysis.

Unstable routing is one of the problems that routing protocols face. An unstable routing might cause frequent routing calculation in a network. This often creates routing loops. *Source routing* and *traceroute* packets are often used for locating routing loops [40]. However, this is not the scope of this research, as the thesis focuses on OSPFv3. Link state routing protocols like OSPF are prone to routing loops as any change in the network is immediately flooded throughout the network.

2.6 Summary

Based on the above observations, it can be said that most of the works to discover IPv6 topology has been done on layer 3. The two papers which focused on both layer 2 and layer 3 discovery in IPv6 networks were [1] and [2]. The problem of huge address space of IPv6 mainly occurs in the case of layer 2 topology discovery as ICMPv6 packets in the entire subnet can generate a lot of traffic. Hierarchical method [1] can be a solution to this problem. However, the main problem of hierarchical structure is that it requires one probe engine in every subnet. As a result, the system gets slow and generates a lot of traffic. Moreover, this is not always possible to keep a probe seed in every subnet. The authors of papers [1] and [2] used neighbour discovery protocol and SNMP for layer 2 discovery. For layer 3 discovery, they used *tracerouting*. For layer 2 discovery, the use of SNMP-MIB can be a good solution. Different geometric solutions are also described in IPv4 topology discovery.

Based on the overall analysis, it can be said that, for layer 3 topology discovery, *tracerouting*, *source routing* and routing protocols are the key mechanisms that have been used. However, one of the problems of *source routing* is that it takes a long time to discover the network topology and generates a large amount of traffic. As we saw in [3], their algorithm took eight weeks to discover approximately 2,420 routers. This happens because of the probing space explosion that was discussed in Section 2.4. The authors in [5] addressed this probing space issue and tried to minimize redundant *source routing* packets. The authors in [6] also addressed this issue and introduced a seed selection method using ICMPv6. In [4], the authors combined *source routing* and routing protocols to reduce the router alias problem and the crosslink problem.

The observation is that *source routing* should not be the sole solution for layer 3 IPv6 topology discovery. A proper seed selection method to reduce probing space is necessary while using *source routing*.

Another problem in *tracerouting* or *source routing* is the initial seed selection (i.e., selection of the initial targets). If the system does not have any previous idea about the whole address space, it is difficult to even start the *tracerouting* algorithm and random selection of addresses can lead to incomplete discovery [3][6].

Section 2.5 showed how *source routing* can be a solution to network management issues. *Source routing* can be used to identify error location and instable routing. If the network operator wants to identify the location of a possible error in the network, *source routing* packets can be generated to locate the place. Moreover, both *traceroute* and *source routing* are useful mechanisms to locate the possible location of packet loss, measure average response time and delay in the network.

As will be explained in Chapter 3, this thesis proposes an IPv6 topology discovery algorithm which also reduces the redundant probing space of *source routing*. We have assumed that the IPv6 network is running OSPFv3 as its routing protocol. The topology discovery of the IPv6 network uses a routing protocol based approach as discussed in Section 2.1.3 and 2.2.3, and the topology information is then used to provide an efficient solution that reduces the redundant probing or probing space for *source routing*.

Chapter 3

A Source Routing Based Algorithm for Efficient Probing Space for IPv6

3.1 Overview

Source routing is an important topology discovery tool. Apart from that, it is also useful for management purposes, ease of trouble location, link cost analysis, etc. It also provides efficient ways to solve the router alias problem and the crosslink problem as discussed in Chapter 2. Moreover, it is also an important tool to discover the IPv6 addresses of the router interfaces while OSPF gives only the subnet information of each link. However, *source routing* has a probing space explosion problem as discussed in Chapter 2. This chapter introduces a probing space reduction mechanism for *source routing* to make it more efficient for both topology discovery and network management.

Routing protocol based topology discovery is a passive discovery approach, as the probing node listens to all the packets that traverse through the link and builds the topology from the obtained information. Discovering network information based on routing protocols has been one of the most widely adopted methodologies in IPv4, as was discussed in Section 2.1.3. However, topology discovery in IPv6 is still a new research field and little research has been conducted on routing protocol based discovery. OSPFv3 is the IPv6 implementation of OSPF, which is currently the most widely deployed routing protocol in the field.

Deploying an approach that makes use of both *source routing* and routing protocol has several advantages. Specifically, combining the information achieved from OSPF and *source routing* can increase the efficiency of the discovery algorithm in terms of time, accuracy and coverage. The authors in [4] combined both the *source routing* and routing protocol information to solve the crosslink and router alias problems of *source routing*. This thesis proposes an algorithm that addresses the probing space explosion issue of *source routing* in networks running OSPFv3 protocol. The objective is to discover the network topology while reducing the amount of redundant probing in *source routing*. This can be achieved by using the topology information from OSPFv3.

As will be seen, the OSPFv3 discovery method provides us with the complete view of the IPv6 network, including the connectivity between the nodes and the associated cost of the links. OSPF uses the well-known Dijkstra's Shortest Path First (SPF) algorithm, which determines the shortest path from each node to every other node of the network. The main idea of the proposed approach is to use the shortest path information between the nodes to reduce the redundant probing of *source routing*. Although the proposed algorithm does not focus on the router alias or cross link problems, the probing space reduction method can be adopted to address those issues in IPv6 discovery.

The remainder of this chapter is organized as follows. The background of OSPFv3 based discovery and the probing space explosion of *source routing* are reviewed in Section 3.2. Section 3.3 presents the advantage of the probing space reduction algorithm in terms of network management. Section 3.4 discusses the OSPFv3 stack of IPv6 in details and describes the Link-State Advertisements (LSA) structures and the information inside these LSA structure to build the topology. Section 3.5 depicts the architecture of the proposed algorithm. Section 3.6 presents the algorithm. Section 3.7 provides a detailed explanation of the proposed algorithm with an example. Section 3.8 provides the overall design and realization of the algorithm. Finally, Section 3.9 is a brief summary of this chapter.

3.2 Background of Probing Space Reduction

Source routing and OSPFv3 are the two main discovery methods to retrieve the topology of IPv6 networks. In OSPFv3, routers periodically share topology information with each other to maintain and update their routing tables and Link State Database (LSDB). Based on the information from the routing tables, routers make forwarding decisions for all packets in the network. Routers exchange the information in the form of link-state advertisements which are packets that contain the connectivity, cost, interface id, prefix information, etc. The idea of OSPF based discovery is to capture these LSAs from the link and process the information inside to build the complete topology of the network.

Source routing is an improved *traceroute* based algorithm which can increase the coverage of the topology discovery. In *traceroute*, routers send ICMP packets with increased hop limit to a destination. If the hop limit mentioned in the IP/IPv6 header is reached, each intermediate router to the destination sends back an ICMP *Time Exceeded Message* to the source of the packet [26]. From these *Time Exceeded Messages*, the path to the destination and also the IPv6 address of each intermediate router can be retrieved. *Source routing* applies a similar concept except that the operator can specify the path to the destination. Moreover, in *source routing*, multiple intermediate routers to the destination can be specified in the header.

Source routing is helpful in obtaining the IPv6 addresses of the router interfaces. As discussed in Chapter 2, in *source routing* all the intermediate routers respond with an ICMP *Time Exceeded Message.* The source address of these messages is the IPv6 address of the responding interface of that intermediate router. The probing machine, from where the *source routing* packets were generated, knows the full IPv6 address of the responding interface of that intermediate router from the *Time Exceeded Message*. However, the selection of the responding interface of an intermediate router is different for ICMP and ICMPv6. In IPv4, the source address of an ICMP *Time Exceeded Message* can be any of the gateway's address of the intermediate router [45]. Thus, multiple probing packets are sent to retrieve the IP addresses of all the interfaces of an intermediate router. However, in IPv6, the source address used by ICMPv6 responses (*Time Exceeded Message*) is defined as either: (1) the unicast address to which the *traceroute* packet is destined or (2) an address belonging to the responding node (intermediate router) that will be most useful in diagnosing the error [3]. According to IETF, the source address may be selected in such a way that this would lead to a more informative choice of address [29]. *Source routing* in IPv6 topology discovery is used to retrieve as many IPv6 addresses as possible. This means, the interfaces of an intermediate router are probed by *source routing* packets to discover as many IPv6 addresses as possible. The *source routing* based algorithm proposed in this chapter ensures probing the maximum number of interfaces possible of all the routers in a network while reducing the number of *source routing* packets (probing space). This ensures the discovery of maximum IPv6 addresses.

However, as discussed in section 2.2.2.1, probing from multiple sources will discover more IPv6 addresses as the routers will choose different interfaces to respond back to different probing sources with ICMP Time Exceeded messages. Figure 3.1 shows how the information of multiple interfaces of a router can be discovered by two probing sources. A source routing packet from probe engine 1 traversing through *A-C-B* will discover the IPv6 address of interface 1 of node *C*. Another source routing packet traversing through *B-C-A* would discover the address of interface 2. However, the number and location of probing sources is important to maximize the discovery as discussed in Chapter 2.

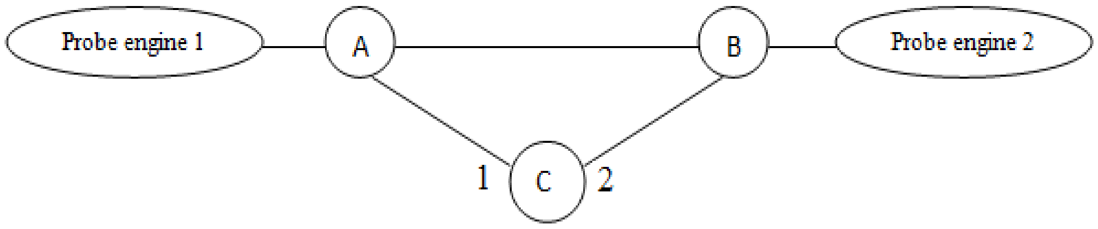


Figure 3.1 Discovering IPv6 address by multiple probing sources

Moreover, as discussed in Chapter 2, *source routing* is also a useful mechanism for network management and troubleshooting. Section 3.3 presents how the *source routing* based algorithm can be an efficient solution for network management.

Source routing has some other drawbacks in terms of security. The basic idea of *source routing* is that a sender can decide the routing path for a specific packet. *Source routing* in IPv6 uses the type 0 routing header where a sender can include multiple IPv6 addresses for the packet to traverse through these nodes. It also allows a sender to include the same address more than once in the same routing header. Therefore, a packet can be constructed to oscillate between two routers. The oscillation phenomenon allows a stream of packets from an attacker to be amplified along the path between two remote routers, which could be used to cause congestion along arbitrary remote paths and hence act as a

Denial-Of-Service (DOS) mechanism [30]. This security concern for *source routing* was present in IPv4 too, which is the reason why most of the ISPs keep the *source routing* capability of their routers disabled. The availability of *source routing* in IPv6 routers will likely follow the pattern of its forebearer [3]. However, as IPv6 is still in its early phase, *source routing* is still considered as a vital method in IPv6 topology discovery. Moreover, for campus network, where the security issue in terms of DOS might not be a major problem, the network administrators can turn on the *source routing* capability of their routers to have a more efficient monitoring on their network. This is one of the reasons why research efforts are still being conducted on IPv6 *source routing*.

All the existing approaches to reduce the probing space of *source routing* are already discussed in Chapter 2. In this thesis, the topology information retrieved from OSPFv3 packets is used to find an efficient solution for the probing space explosion problem of IPv6 *source routing*. Before moving to our algorithm, the OSPFv3 stack for IPv6 needs to be discussed as OSPFv3 has a lot of differences with OSPFv2 (Open Shortest Path First for IPv4). The following section discusses the advantages of probing space reduction in terms of network troubleshooting.

3.3 Advantage of Probing Space Reduction in Terms of Network Troubleshooting

As discussed in Section 2.5, *source routing* is an important network management mechanism for troubleshooting. Network operators often face challenges in finding problems in the network. Generating ICMP packets is a useful solution for locating network problems. However, constant monitoring on the network and generating *source routing* packets blindly is not an efficient solution. In this research, we tried to reduce the probing space of *source routing* which is a useful solution for locating failure points in the network. For a large network, it often gets difficult to locate the exact point of failure. It is not possible to generate network wide *traceroute* and *source routing* packets because of the probing space issue of *source routing*. Chapter 4 presents experiments and shows that the probing space is greatly reduced. For example, an OSPF area size of 70 routers might have a probing space of only 70 to 100 to probe the interfaces of all the nodes in the network. A network operator can generate all the *source routing* packets to locate all the possible error in the network. If the network operator knows the topology of the network, *source routing* packets can be generated according to the algorithm more efficiently. Even if the operator knows the topology, a network might go through subsequent changes at any point in time. A network problem may be caused from different reasons like link failure, node failure, etc. Also, OSPF routers may lose adjacency if, for some reasons, a node does not get any response to its HELLO packets before the dead interval (set to 40 seconds by default) which is usually four times than the HELLO interval.

Moreover, as discussed in Chapter 2, the transmission quality of a network is measured by packet loss, delay and delay variation [43]. *Source routing* can measure the elapsed time between transmission and reception of an ICMP packet along with delay and loss characteristics of a path. The *source routing* algorithm provides an efficient solution for network wide measurement of transmission quality. For all these network phenomena, we believe a network operator can generate *source routing* packets much more efficiently to detect any errors or changes in the topology.

3.4 Open Shortest Path First for IPv6 (OSPFv3)

OSPF is a link-state routing protocol that uses Dijkstra's SPF algorithm to determine the shortest path between any two nodes. RIP uses hop count as it's routing metric which is not effective for large networks. OSPF divides the autonomous system into areas to reduce the size of the LSDB and the processing overhead for routers, which makes the network more scalable.

A principal advantage of OSPF in terms of topology discovery is that the link-state routing protocol can create a complete view or topology of the network by gathering information from all the other routers [33]. OSPF learns route information by exchanging LSAs. Each router maintains a LSDB that depicts the topology of the network. The

LSDB of all the routers in an OSPF area is the same after the system reaches the steady state.

The fundamental mechanisms of OSPFv2 (flooding, designated router election, area support, shortest path first calculation, etc.) remains unchanged in OSPFv3 [34]. However, there are several important changes over OSPFv2 that makes the functionality of OSPFv3 different than OSPFv2. A brief description of the main differences between the two protocol stacks is presented here:

- OSPFv3 uses the link-local IPv6 address of the routers as the source and next-hop addresses. This address always begin with FF80::/10 [35].As this address is locally significant, these packets will never be routed by a router. The host portion of the link-local address is actually the host portion of the global IPv6 address of a node. From the perspective of IPv6 topology discovery, these link-local IPv6 addresses cannot be seen by a distant node (or the discovery algorithm) as these addresses are locally significant. This means that the host portion of the IPv6 address cannot be retrieved by an OSPF based discovery algorithm. A discovery algorithm using OSPFv3 can only retrieve the prefix information of each node from intra-area prefix LSAs. One of the motivations of this research was to introduce an efficient *source routing* mechanism that would discover as many IPv6 addresses as possible as they cannot be discovered by the OSPFv3 based algorithm. The following figure shows an example of IPv6 address with a 64 bit prefix.

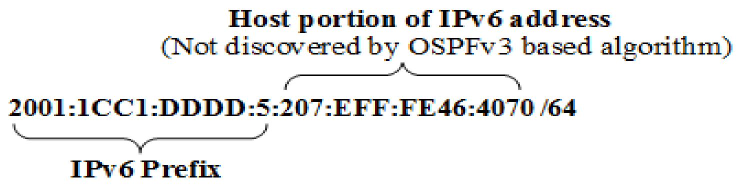

Host portion of IPv6 address
(Not discovered by OSPFv3 based algorithm)
2001:1CC1:DDDD:5:207:EFF:FE46:4070 /64
IPv6 Prefix

Figure 3.2 Incomplete IPv6 Address Discovery by OSPFv3 Based Algorithm

- The term "subnet" in OSPFv2 is replaced with the term "link" in OSPFv3. In IPv6, multiple subnets can be assigned to an interface. Moreover, in OSPFv3, there is no "same subnet" requirement to form neighbor adjacencies [28]. This means that two routers or nodes can still communicate even if they belong to different subnets.

- The router LSA and the network LSA do not carry any prefix information anymore. Prefix information is carried by a new LSA type called intra-area prefix LSA [34]. This makes the flooding of OSPFv3 packets more scalable than OSPFv2. Router and network LSAs are only flooded when information related to SPF calculation is changed. Any changes in the prefix information are carried out by intra-area prefix LSAs, which prevent unnecessary SPF calculation.
- IPv6 uses scope-based addressing scheme. As the addressing scheme is scope-based, OSPFv3 also uses scope-based flooding. The three flooding scope of OSPFv3 are: i) link-local, ii) area level, and iii) AS level. In this research, we have focused on the area level discovery and processed all the LSAs that have area level flooding scope. Apart from carrying the LSA function code in the LSA header, OSPFv3 also carries the flooding scope of the LSA that is embedded in the header [28].
- Neighbor specific information is carried by a new LSA type called link LSA [34]. OSPFv2 used to flood this information with router and network LSAs throughout the area, whereas OSPFv3 floods this information using link LSAs only on the links shared by two nodes. It reduces a lot of unnecessary traffic in the network. This is a problem in terms of IPv6 topology discovery. As the local information is only shared between the two neighbors, a distant node or a discovery algorithm cannot retrieve this information.
- OSPFv3 supports multiple instances per link [28]. This means that several autonomous systems running OSPF can share a common link. It adds an OSPF instance id into the OSPF packet header to distinguish multiple instances. Multiple instances of OSPFv3 is not within the scope of this research.
- There are several other important differences like removal of OSPF authentication, handling unknown LSA types, etc., which are also beyond the scope of this research.

Two new types of LSA have been introduced in OSPFv3: link LSA and intra-area prefix LSA. Although the basic functionality of the other packets is the same as OSPFv2, there are differences in their functionality and carried information as we discussed earlier. All the OSPFv3 LSA types with their function codes and corresponding OSPFv2 LSAs are presented in Table 3.1.Among the LSAs mentioned in Table 3.1, the AS-External LSA has the autonomous system level flooding scope and the link LSA has the link local flooding scope. All the other LSA types have the area level flooding scope [34]. This research focuses on the area level discovery and therefore only deals with router LSA, network LSA, intra-area prefix LSA and link LSA. We also processed the information of link LSA to learn about the neighbour routers.

Table 3.1 OSPFv3 LSA Types [28]

LSA Type (OSPFv3)	**LSA name (OSPFv3)**	**LSA name (OSPFv2)**
0x2001	Router LSA	Router LSA
0x2002	Network LSA	Network LSA
0x2003	Inter-Area Prefix LSA	Network Summary LSA
0x2004	Inter-Area Router LSA	ASBR Summary LSA
0x4005	AS-External LSA	AS-External LSA
0x2004	Group Membership LSA	Group Member LSA
0x2007	Type-7 LSA	NSSA External LSA
0x0008	Link LSA	N/A
0x2009	Intra-Area Prefix LSA	N/A

A short discussion on each of these LSA types is provided below:

Router LSA. Unlike OSPFv2, in OSPFv3, the router LSAs do not include any prefix information. A router LSA includes information about the originating router, neighbour router and the link information attached to that router. In short, router LSAs provide the overall connectivity of the network. Each link is identified by the interface id of the originating router and the interface id of the neighbour router. Router LSAs also carry the cost information of each link between the originating router and the neighbour router. The interface type field determines the type of the interface of a router. An interface type 1 means the associated link is a point-to-point link and an interface type 2 means the associated link is a transit link [34][28].

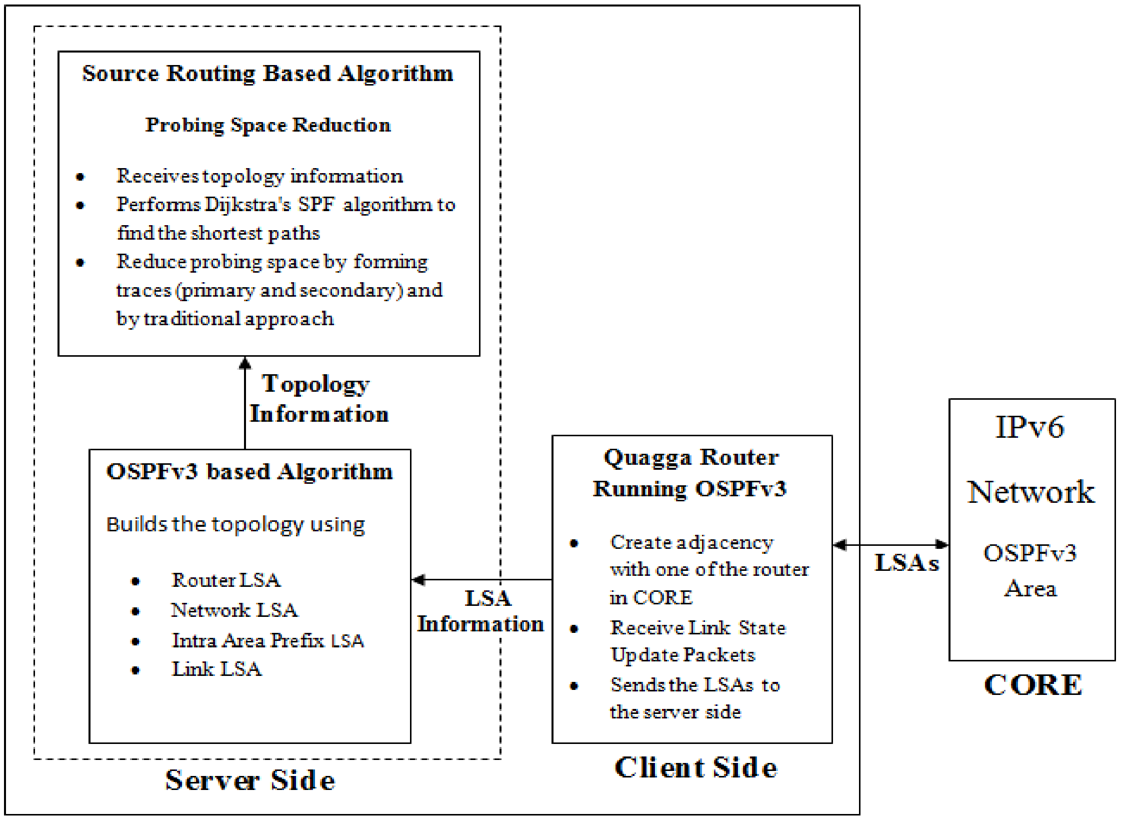
Network LSA. Network LSAs carry the information about the transit links. Network LSAs carry the information about all the attached routers in a link. Network LSAs also identify the Designated Router (DR) of a link. Both the router LSA and network LSA are used together to depict the information about point-to-point links and transit links. Network LSA is originated by the DR of a link and the link state id is always the interface id of the DR's interface to that link [28][34].

Intra-Area Prefix LSA. A router uses intra-area prefix LSA to advertise one or more IPv6 prefixes associated with either the router LSA or a network LSA [28]. This association with the router LSA and network LSA is identified by the Referenced LS (Link State) Type, Referenced LS Id and Referenced advertising router. If the prefix LSA is associated with a router LSA, then the Referenced Link State ID of the Intra-Area Prefix LSA will be "0" and the Referenced Advertising Router is the router id of the originating router. If the prefix LSA is associated with a network LSA, the Referenced Link State ID is the interface Id of the DR and the Referenced Router is the router id of the links DR.

Link LSA. Link LSAs are only shared between two routers. Therefore, link LSAs are locally significant and do not traverse beyond one link. Thus, we cannot receive any link LSA from a distant router. Only link LSAs carry the full IPv6 addresses of router interfaces [34]. Therefore, the full IPv6 address cannot be discovered by an OSPF based topology discovery approach.

3.5 Architecture of OSPFv3 based Topology Discovery and Source Routing based Algorithm

This section presents the architecture of the proposed approach that uses both OSPFv3 and *source routing*. The algorithm is a combination of both passive and active discovery techniques. In a passive technique, a probing node only listens to all the information going through its link. The OSPF based algorithm is a passive algorithm, as it only listens to the LSAs to build the network topology. An active technique generates packets in the network to retrieve the topology information. The *source routing* based approach is an active one, as it generates both the *traceroute* and *source routing* packets to retrieve information about the links and nodes. In this research, the *source routing* based algorithm relies on the information obtained from the OSPF based algorithm to introduce an improved discovery algorithm that reduces the probing space of *source routing*. A brief overview of the architecture is shown in Figure 3.3.



Source Routing Based Algorithm
Probing Space Reduction
Receives topology information
Performs Dijkstra's SPF algorithm to find the shortest paths
Reduce probing space by forming traces (primary and secondary) and by traditional approach
Topology Information
OSPFv3 based Algorithm
Builds the topology using
Router LSA
Network LSA
Intra Area Prefix LSA
Link LSA
LSA Information
Quagga Router Running OSPFv3
Create adjacency with one of the router in CORE
Receive Link State Update Packets
Sends the LSAs to the server side
LSAs
IPv6 Network
OSPFv3 Area
CORE
Server Side
Client Side

Figure 3.3 Architecture of Topology Discovery and Source Routing based Algorithm

As shown in Figure 3.3, the server side runs both the OSPFv3 based algorithm and the *source routing* based algorithm. Quagga is an open-source routing software suite that provides the implementation of routing protocols [32]. Quagga receives the LSA information from the IPv6 network and sends the information to the server side for computation. The network is configured in an open source emulator called CORE [31]. The OSPFv3 based algorithm in the server side processes all these LSAs and builds a complete view of the topology. This research focuses on an area level topology discovery. Therefore, the OSPFv3 based algorithm only processes four types of LSAs: router LSA, network LSA, intra-area prefix LSA and link LSA. A client-server architecture is adopted between Quagga and the server. A client program is put inside the source code of Quagga. A group of LSAs come together inside a link state update packet. The client program is placed in the source code, where the link-state update packets are received. The client side sends these LSAs to the server side, where these LSAs are processed to build the topology. The OSPFv3 based algorithm in the server side takes only a few seconds to discover the full topology of the network.

After building the full connectivity information, the OSPFv3 based algorithm sends the topology information to the *source routing* based algorithm. As OSPFv3 uses Dijkstra's SPF algorithm, the *source routing* based algorithm can run SPF to know the shortest path from each node to every other nodes in the network. The *source routing* based algorithm provides an efficient solution for *source routing* reducing the probing space. The probing space reduction is performed in two steps:

- Probing space reduction by forming traces
- Traditional approach

Both of these steps are presented in Section 3.6. Also, a detailed discussion with flowcharts on both the OSPFv3 based algorithm and the *source routing* based algorithm is presented in Section 3.7.

3.6 Topology Discovery Algorithm to Reduce Probing Space of Source Routing

In this algorithm, we assume that the full IPv6 addresses of at least two initial routers are known. These addresses are termed as initial seeds. These routers are the edge routers (as far as possible from quagga) of the OSPFv3 area. The following summaries the list of symbols used in our algorithm.

Symbols:

T_{router} = All the discovered router ids by OSPF.

T_{link} = All the discovered links between routers by OSPF.

T_{prefix}= All the discovered prefixes by OSPF.

$T_{interface}$= All the discovered interfaces (interface id) by OSPF.

T_{metric}= Associated cost of each link found by OSPF.

R_{trace}= Set of routers found by *traceroute*, where $R_{trace} \in T_{router}$

R_{link}= Set of links found by source route, where $R_{link} \in T_{link}$

$trace_n$= Set of routers in the n^{th} trace.

E = Set of Addresses of the initial seeds.

The detailed steps for both the OSPFv3 based algorithm and source route based algorithm are described below:

OSPFv3 based Algorithm

In the following, the first four steps includes running the client-server program, starting the Quagga daemons and the formation of adjacency between the Quagga router and a router from the emulated network. Lastly, step 5 is responsible once the server starts receiving packets.

```
1. Run the server side program that will receive LSA information from the client
   side.
2. Start the daemons of Quagga and create adjacency with one of the routers in
   backbone area.
3. Both the Quagga router and the router in the OSPF area start exchanging
   OSPFv3 packets.
4. Start receiving all the OSPF packets in the server side via the client side placed
   in Quagga.
5. if (received packet is a router LSA)
       {
       save the connectivity information, including:
               • router id and neighbor router id in $T_{router}$
               • discovered links in $T_{link}$
               • interface id and neighbor interface id in $T_{interface}$
               • cost of each link in $T_{metric}$
       }
   else if (received packet is a network LSA)
       {
       save the connectivity information and properties of the transit links
       including:
               • designated router of the transit link.
               • all the router id in $T_{router}$
               • discovered links in $T_{link}$
       }
   else if (received packet is an intra area prefix LSA)
       {
               • Find associated router or network LSA
               • save IPv6 prefixes corresponding to the router id and
                 interface id in $T_{prefix}$
       }
   else
       Discard the packet.
```

The OSPFv3 based algorithm provides the full connectivity information of the topology. A detailed discussion on the algorithm is provided in Section 3.7.1.

Source routing based Algorithm

The first three steps represents the formation of primary traces. Step 4 represents the formation of secondary traces and the probing space calculation by traditional approach.

```
1. Traceroute all the initial seeds element of $E$
2. Save the routers found in each traceroute result in $trace_n$. These are the primary
   traces,  where $trace_n$ is the set of routers found in the $n^{th}$ trace and $trace_n \in T_{router}$
3. Save all the new found links in $R_{link}$, where $R_{link} \in T_{link}$
4. for each trace in $trace_n$
   {
   for each router $R$ in $n^{th}$ trace
           {
               • perform SPF on $R$ and find all the paths to the routers of other
                 traces
               • save the path with the most number of unknown links as a new
                 trace $trace_n$
               • update $trace_{n.}$
               • perform source route between the two edges of this new trace
                 and update  the probing space.
               • save each new found link in $R_{link}$.
           }
      for all the primary and secondary traces
           {
       if(no new trace is formed)
             {
               • Stop formation of traces and proceed to traditional approach
             }
           }
   }
   if($R_{link} \neq T_{link}$)
     {
      find the probing space by traditional approach:
               • for each link $L$, where $L \in T_{link}$ and $L \notin R_{link}$, generate a random
                 host portion of IPv6 address
               • append it with the prefix information from $T_{prefix}$
               • perform traceroute and then source route to these nodes
               • update the probing space
      }
```

3.7 Explanation of the Algorithm

3.7.1 OSPFv3 based Algorithm

The following flowchart provides the functionality of the OSPFv3 based algorithm.

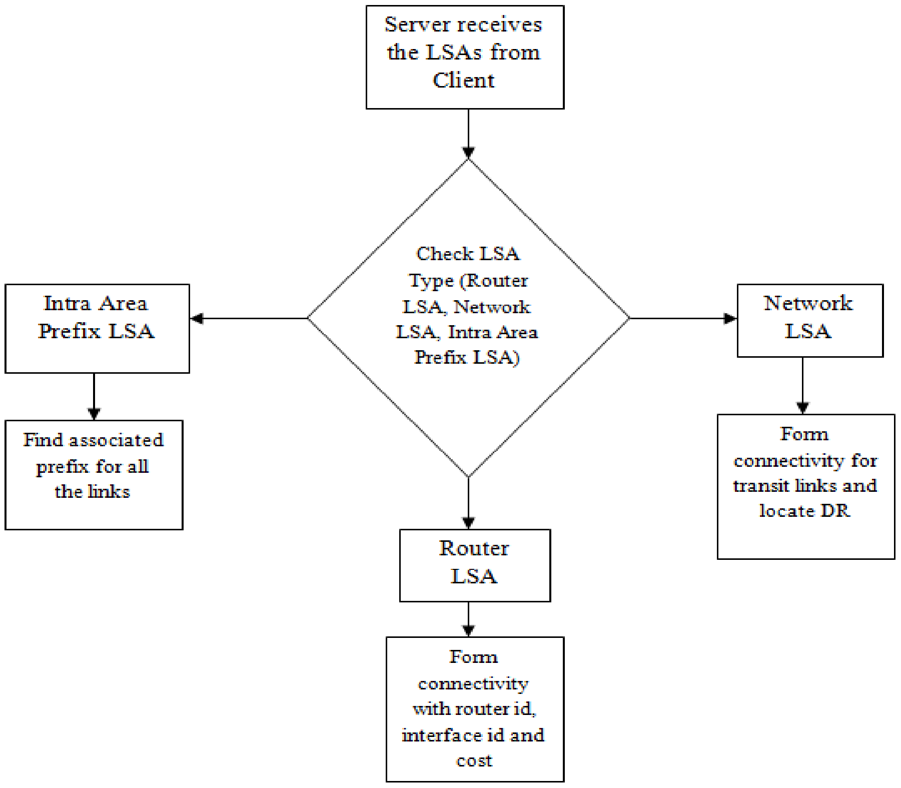

Server receives the LSAs from Client
Check LSA Type (Router LSA, Network LSA, Intra Area Prefix LSA)
Intra Area Prefix LSA
Find associated prefix for all the links
Network LSA
Form connectivity for transit links and locate DR
Router LSA
Form connectivity with router id, interface id and cost

Figure 3.4 Flowchart of OSPFv3 based Topology Discovery Algorithm

The OSPFv3 based algorithm discovers the full connectivity information of the network. As the algorithm is an area discovery algorithm for OSPF, only three types of LSAs are processed. As shown in Figure 3.4,the router id, neighbor router id, interface id, neighbor interface id and the link cost are discovered from the router LSA. This information gives the overall connectivity of the network. The network LSA provides the information of transit links along with the information of the DR. Prefix information of all the links are known from the intra area prefix LSA. Thus, from the OSPF, we already have the connectivity information of the topology, the prefix information and the cost of each link. Figure 3.5 provides a brief example of the information discovered by the OSPFv3 algorithm. If T_{link} is all the link information, then in Figure 3.5 T_{link} will consist of *J-I*, *I-M*, *I-N* and *M-N*. *J*, *I*, *M* and *N* represents the router ids of the routers.

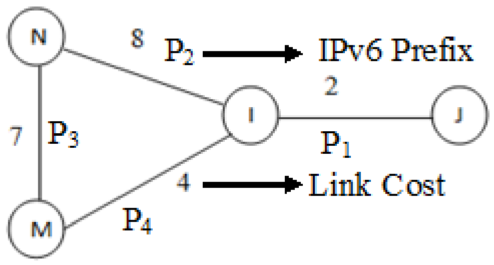


Figure 3.5 Example of the Connectivity Information obtained from OSPFv3

For example, assume that the cost of links *J-I*, *I-M*, *M-N* and *I-N* are 2, 4, 7 and 8, respectively. Then the OSPF topology discovery will generate the connectivity information in the following manner:

- *J* is connected to *I* with associated cost 2 and IPv6 prefix P_1
- *I* is connected to *M* with associated cost 4 and IPv6 prefix P_4
- *M* is connected to *N* with associated cost 7 and IPv6 prefix P_3
- *I* is connected to *N* with associated cost 8 and IPv6 prefix P_2

As the connectivity information of the routers with the associated cost is discovered, it is possible to determine the shortest path from each router to every other routers of the OSPF area with Dijkstra's SPF algorithm.

3.7.2 Source routing based Algorithm

The *source routing* based algorithm ensures probing the maximum number of nodes and links in the network while reducing the probing space of *source routing*. In this research, we assume that we know the full IPv6 address of at least two edge routers as the initial seeds. The *source routing* based algorithm *traceroute* to these initial seeds and forms primary traces. Each router of these traces performs the Dijkstra's SPF algorithm to find the shortest path to the routers of other traces. A new trace (called secondary trace) is formed if a router finds a shortest path that would give the information of the most number of links that are not probed yet. Thus, the path that has the most number of links that are not yet probed will be selected as a new trace (secondary trace). The two edges of a secondary trace become a new probing pair for *source routing*. Probing pair is the pair of an intermediate node and a destination node for *source routing*. The following flowchart shows the functionality of the *source routing* based algorithm.

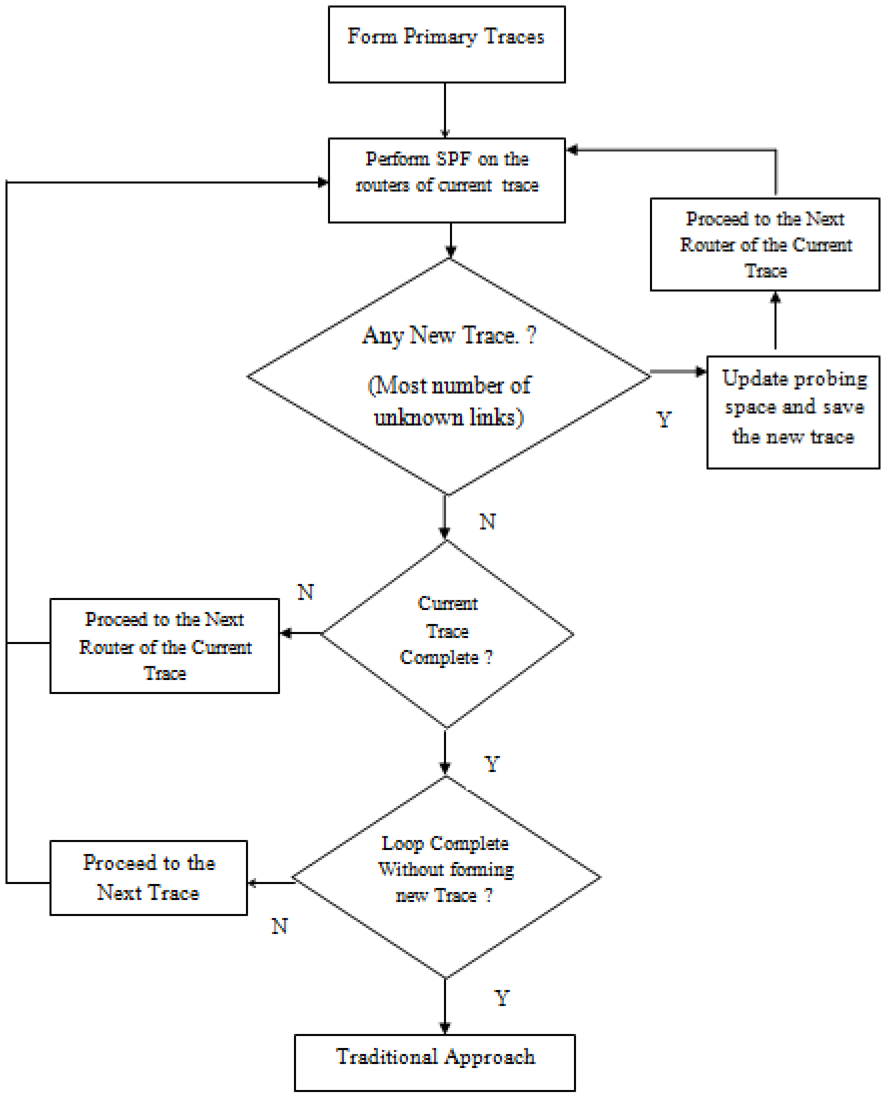
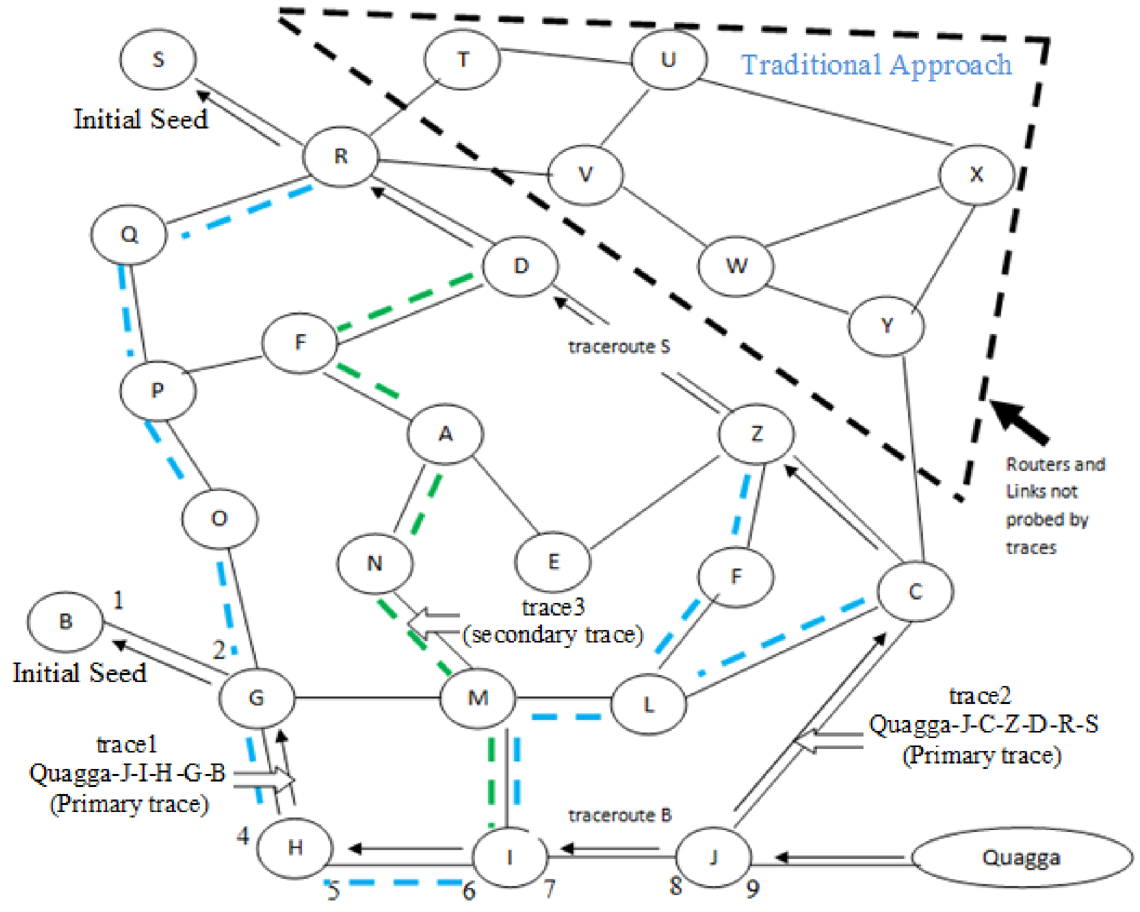

Form Primary Traces
Perform SPF on the routers of current trace
Proceed to the Next Router of the Current Trace
Any New Trace. ? (Most number of unknown links)
Y
Update probing space and save the new trace
N
Current Trace Complete ?
N
Proceed to the Next Router of the Current Trace
Y
Loop Complete Without forming new Trace ?
N
Proceed to the Next Trace
Y
Traditional Approach

Figure 3.6 Flowchart of Source Routing based Algorithm

As can be seen from Figure 3.6, SPF is performed on the routers of each trace (primary and secondary) to form new traces. The pair of routers at the two ends of a new trace is the new probing pair for *source routing.* This is why the probing space is updated, each time a new trace is formed. After running SPF in all the routers of a current trace, the algorithm moves to the next trace to do the same. The algorithm keeps performing SPF in all the primary and secondary traces until it cannot form any new trace. This means that all the links between the primary traces are known and probed by the algorithm. If no new trace is formed for an entire loop of traces, the algorithm moves to the traditional approach to calculate the probing space for the rest of the



⟶	traceroute path
trace$_1$ & trace$_2$	Primary traces formed by traceroute
🟦 🟦 🟦	Shortest paths to nodes from other traces
🟩 🟩 🟩	New trace formed giving the most number of unknown links from *I*
■ ■ ■	Routers & Links that cannot be probed by forming traces

Figure 3.7 Reduction of Probing Space

links in the network. This means that for all the existing traces (primary and secondary), if no new trace is formed, then the rest of the links will be probed by the traditional approach. For n interfaces, using the traditional approach would generate a probing space of n(n-1) as was discussed in Chapter 2.

In Figure 3.7, assume that nodes *B* and *S* are two initial seeds.

The following explains the steps of the algorithm in detail with an example based on Figure 3.7:

Step 1. At first, we perform the *traceroute* operation to the initial seeds *B* and *S*. The arrow signed paths are the traveled paths to these nodes. Because of these *traceroute* messages, all the intermediate nodes will send back an ICMP *Time Exceeded message* to the source node Quagga. As discussed earlier, Quagga is a routing software suite that emulates a router with routing protocols. Therefore, Quagga works as another router exchanging the same routing information like the other routers in the OSPF area. The ICMP *Time Exceeded messages* will have the complete IPv6 address of the corresponding routers as their source address. This way, the interface information of these routers are known. The result of these two *traceroute* operations will be saved in $trace_1$and $trace_2$, where $trace_1$ is the list of routers that was found by the *traceroute* to *B* and $trace_2$is the list of routers from *traceroute* to *S*.$trace_1$and $trace_2$are the primary traces formed by the initial seeds. We save the links between these routers in R_{link}, where R_{link} is the set of links found by *source routing*.

Step 2. Dijkstra's SPF algorithm is performed on all the routers of these traces and form new traces. These traces are called the secondary traces. We continue performing the SPF algorithm on the newly identified traces as well and keep forming more secondary traces. For example, as shown in Figure 3.7, we start with $trace_1$ first. For a particular router from $trace_1$, we figure out the shortest paths to all the routers from other traces. We calculate how many links can be probed if a source route packet traverses between two of these nodes. For a pair of nodes, if we can find the largest number of links that are not yet probed (links that are not in R_{link}), we select those two nodes as our probing pair. For example, for node *I*, there are a number of shortest paths to the routers from $trace_2$ (blue dotted lines). If node *I* has a shortest path to *Z* which consists of *I-M-L-F-Z*, we can probe four new links if we perform a source route between node *I* and node *Z*. But there is another path from node *I* to node *D* marked as the green dotted lines in the figure, which can probe five new links. This means that if a *source routing* packet is generated, keeping nodes *I* and *D* as intermediate router and destination router, it would traverse through the most number of links that are not yet probed. Thus, we save this path information as a new trace referred to as $trace_3$. This trace is termed as secondary trace. We perform the same thing for all the routers from $trace_1$ and build new traces.

Step3. After calculating the SPF on all the routers from $trace_1$, we move to the next trace and perform the same process. The new traces that are found from this process will also be included in this calculation. As we are done with $trace_1$, we move to $trace_2$ and perform the same procedures outlined in step 2. But this time, as we formed a new trace ($trace_3$) in step 2, we perform SPF on the routers of $trace_2$ for all the other routers of both $trace_1$ and $trace_3$. We perform this procedure repeatedly on all the routers and try to minimize the number of unknown links. When, for all the existing traces, no new trace is formed, the algorithm moves to the traditional approach to calculate probing space for the rest of the links (step 5).

Step4. Each time a secondary trace is formed, the algorithm updates the probing space. As we are selecting two nodes as a probing pair (n = 2), the probing space will be increased by the value $n(n - 1) = 2$.

Probing by traditional approach

As we have discussed, the algorithm starts with some primary traces ($trace_1$ and $trace_2$ in this example) and forms secondary traces between these primary traces. Step 1 to step 4 continues to form secondary traces between these primary traces in such a way so that all the links and interfaces are probed. However, this process might leave some nodes, links and interfaces without being probed or discovered. The dotted lines in black encloses the routers and links that were not discovered or probed by step 1 to step 4. For example, the nodes *T, U, V, W, X* and *Y* will not be probed by this process. The reason for this is the links between the nodes *C* and *R*. As we can see that the path *C-Z-D-R* is the shortest path, any packets traversing from *C* to *R* or vice versa will traverse through this path. Therefore, we cannot generate any *source routing* packet that can traverse through these unknown nodes. This phenomena occurs when our known initial seeds are so close to each other and a major region of the network remains without having any initial seeds. For example, if there were another initial seeds in the location of node *U* or *X*, the number of nodes without being probed, would decrease significantly. Another initial seed would generate another primary trace (e.g. *J-C-Y-X-U*) that would discover the links between itself and $trace_{2}$.

Now, as the prefix information of all the links is already known from the OSPFv3 based discovery, a random host portion of these addresses can be generated for these links (links not probed by traces) to create full IPv6 address. If *traceroute* packets are sent to these addresses, the adjacent node will reply with a destination unreachable message to the source which will have the interface information (full IPv6 address) of that router. *Traceroute* can be performed to all the links that were not probed by the traces to discover as many IPv6 addresses as possible. After that, the algorithm calculates the probing space that these newly found interfaces would generate if *source routing* is performed keeping each two of these interfaces as a probing pair. We call this approach in this research as the traditional approach as n interfaces would generate a probing space of $n(n-1)$.

3.8 Design and Setup of the Proposed Algorithm

The design of the algorithm can be divided into two parts:

1. OSPFv3 based topology discovery.
2. Minimizing Redundant Probing of *Source Routing*.

3.8.1 OSPFv3 Based Topology Discovery

The objective of the OSPFv3 based topology discovery was to build a non-intrusive automated topology discovery algorithm that will notify any kind of topology changes instantly, so that the administrator can perform a live monitoring on the network. The algorithm will also update the topology map upon any kind of changes in the network. More specifically, we wanted to build an algorithm that would:

- Constantly monitor the network without being intrusive. This can be achieved by monitoring the exchanges of routing protocol information such as the OSPFv3 LSAs.
- Notify instantly if any changes are observed.

The OSPF routers exchange link state information by LSAs on a periodic basis and update their LSDB and routing table. In a particular area, each router maintains a copy of the LSDB. OSPF allows routers to dynamically learn routes from other routers and advertise routes to other routers. Each OSPF router keeps track of the state of all the various network connections in its area. That means that the LSDB indicates which router can reach other routers and also the subnet information of these routers. If any changes occur in the topology, the LSDB and the routing table of each router in an area are updated. A large autonomous system is divided into small areas to reduce the complexity (i.e. reduces routing traffic and keeps the LSDB in each router small).

To have the up-to-date information about the network, performing a routine check-up of the LSDB and updating the information according to that is important. Any changes in the topology (such as link failure, addition of new nodes, etc.) is unpredictable. As a result, the collection of information has to be done very frequently. Analyzing the LSDB frequently or taking snapshots in a periodic manner is a daunting task. Moreover, if the notification is not live, it might degrade the quality of service as well as create security issues. Moreover, modern traffic engineering requires fast information collection of the network topology in order to perform efficient traffic management, failure restoration, etc.

3.8.1.1 Methodology of OFPF based Topology Discovery

The LSAs that are exchanged between the routers carry the topology information of an OSPF area. One way of building the topology is to capture these LSAs and process the information inside these packets. Our approach to fulfill the objective is to configure a work station that will act like a router and create adjacency with one of the routers in an OSPF area. As it creates the adjacency, it automatically gets included in that area and starts exchanging link state information with the adjacent router. Similar to the other routers in the area, it will also maintain a copy of the LSDB and an up-to-date routing table.

To make the workstation act like a router, Quagga is used in this work. Quagga is an open source software providing implementation of various routing protocols and runs in a Linux platform [32]. We configured Quagga to run OSPFv3. We used an open source emulator called CORE to build our test network which also runs OSPFv3 [31]. We established adjacency between one router of the emulated network and the Quagga router so that the Quagga router falls into the same area of the emulated network and exchange OSPF packets. Figure 3.8 depicts the basic methodology used in this work.

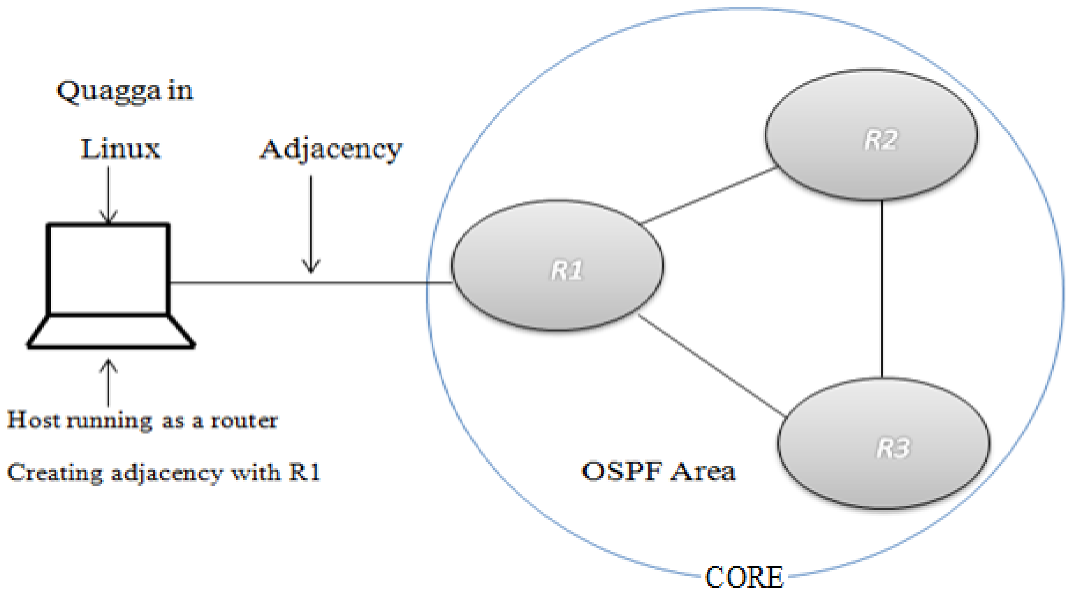


Figure 3.8: Discovering Topology of an OSPF Area

Quagga and the emulator, both running OSPFv3, were configured in two separate Linux machines. The two machines were connected with a Fast Ethernet cable. The emulator CORE is an open source software and has a GUI (Graphical User Interface) that is used to build the network. CORE also uses Quagga for its routing purpose. Each node in the emulator works as an individual virtual machine. Each node was configured with OSPFv3 and the entire configuration was saved as a.*imn* file. The emulator has two modes: configuration mode and execution mode. Once the emulator is in execution mode, the nodes of the network starts sharing the OSPF packets.

Quagga was implemented in a separate machine. Out of the 6 daemons of Quagga, only Zebra and OSPF6d were configured to implement OSPFv3. Once both of these daemons are started, the Quagga router creates adjacency with one of the routers of the emulated network and starts sharing packets. The exchange of each packet was investigated with the protocol analyzer "Wireshark" [36].

As the link-state advertisements starts being exchanged, LSAs are needed to be captured to build the topology. In this research, a client-server architecture was built to receive and process the LSAs. The client side works as a small hook inside the Quagga source code. The client program was placed in the source code, where the link-state update packets are being received. Link-state update packet is one of the five OSPF packets (hello, database description, link state request, link state update and link state acknowledgement) that contains the LSAs. The client program receives these LSAs and sends them to the server side.

The server side runs the algorithm that processes the LSAs and reveals the connectivity between the nodes. Four types of LSAs (router LSA, network LSA, intra-area prefix LSA and link LSA) were processed to retrieve the topology. The server side algorithm thoroughly looks into each packet, stores the information of the packets and constructs the connectivity between the nodes.

The server side identifies each router with its router id. The router id is a 32-bit number that uniquely identify a router. The format of the router id is the same for both IPv4 and

IPv6. A router might have many interfaces connected to the interfaces of other routers or hosts. The server side also retrieves the interface ids from the router LSAs for each other. This gives us the information about the number of interfaces a router has and also the connectivity between two interfaces. The router LSAs also carry the link cost information for each link. The server side retrieves these cost information that are used to perform Dijkstra's SPF algorithm to find the shortest path. Intra-prefix LSAs are responsible to carry the prefix information of each interface. These prefix information of each link is also discovered by the server side. The figure below shows a sample IPv6 network for one OSPFv3 area.

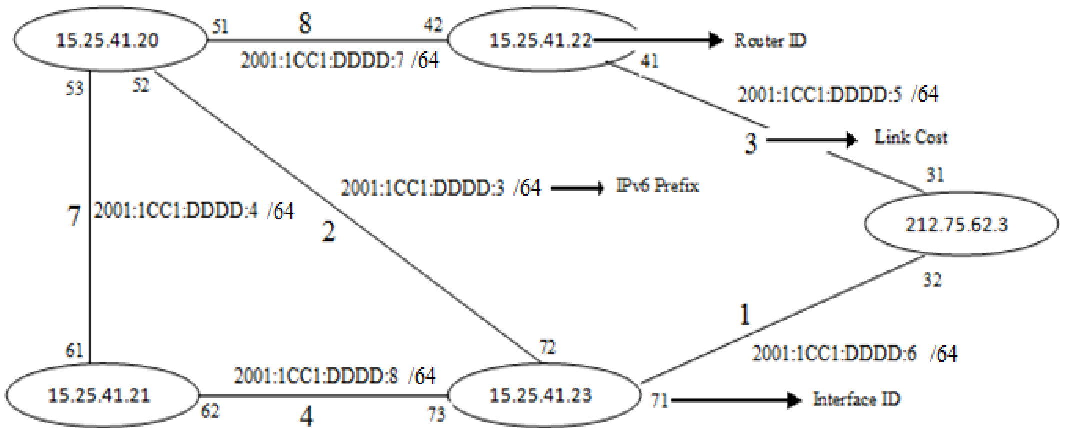


Figure 3.9: Discovered Connectivity Information of an OSPFv3 Area

From the sample topology shown in Figure 3.9, the server side program would discover the topology in the following manner:

Table 3.2: Connectivity Information of a Network Running OSPFv3

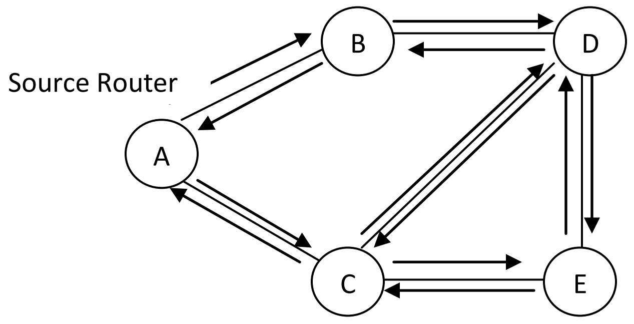
Router ID	Interface ID	Link Cost	Router ID	Interface ID	IPv6 Prefix
212.75.62.3	31	3	15.25.41.22	41	2001:1CC1:DDDD:5/64
212.75.62.3	32	1	15.25.41.23	71	2001:1CC1:DDDD:6/64
15.25.41.23	72	2	15.25.41.20	52	2001:1CC1:DDDD:3/64
15.25.41.23	73	4	15.25.41.21	62	2001:1CC1:DDDD:8/64
15.25.41.20	53	7	15.25.41.21	61	2001:1CC1:DDDD:4/64
15.25.41.20	51	8	15.25.41.22	42	2001:1CC1:DDDD:7/64

As we can see from Table 3.2, the connectivity between all the routers is revealed. For example, the router with router id 212.75.62.3 is connected to the router with router id 15.25.41.22 having the link cost 3. It also shows that the interface id of the two routers are 31 and 41 respectively and the link has a prefix 2001:1cc1:dddd:5. One important thing to note here is that the OSPF based discovery could not discover the full IPv6 address of the router interfaces. The prefix shown here are all 64 bit prefix. Thus, the last 64 bit of the IPv6 address could not be revealed by the OSPFv3 based discovery.

The routers only floods the network information with intra-area prefix LSA throughout the area which is the prefix portion of the IPv6 address. Only the link LSAs exchange the host portion of the IPv6 address. But the link LSA is locally significant and can't be received by a distant node such as Quagga in our case.

3.8.2 Minimizing Redundant Probing of Source Routing

As we know the basic functionality of *source routing*, we implemented the fundamental characteristics of *source routing* within a C program. This part is included in the server side program which was also written in C. We did not generate any *source routing* packet in our network. The effort was concentrated on the output of *source routing*, as if real *source routing* packets were generated. As the concentration of this research was to minimize the probing space, we created a simulation environment to determine the number of probing pairs that *source routing* would generate.



Source Routing From A:

- To D via B
- To B via D
- To D via C
- To C via D
- To E via D
- To D via E
- To E via C
- To C via E
- To B via E
- To E via B
- To C via B
- To B via C

Figure 3.10: Implementation Procedure of Source Routing Based Algorithm

From Figure 3.10, assume that *A* is the source router that generates *source routing* packets. As can be seen, the total number of probing pair is twelve. As the number of probing pair refers to the probing space, the probing space of this small topology is also twelve. That means router *A* has to generate twelve *source routing* packets to probe all the interfaces and links in this topology. In the case of the proposed source route based algorithm, it takes the topology information from the OSPFv3 based discovery and generate probing pairs keeping the number of pairs as minimum as possible. So the algorithm finds the probing space of the topology rather than sending any *source routing* packet. In our algorithm described in Section 3.6, we showed that the unknown links and their interfaces are probed as a form of traces. The algorithm repeatedly finds new traces to generate probing pairs.

Once the algorithm generates all its traces, the probing space or the number of probing pairs these traces would generate is calculated. But unfortunately, these traces will not cover all the links and nodes. Thus, all the links cannot be probed by traces. Links that cannot be probed by traces, are probed using the traditional approach. For each of these links, a random host portion of the full IPv6 address for the link is generated. This host portion of the IPv6 address is then added with the prefix (already known from the

OSPFv3 based discovery) of that link to construct a full IPv6 address for that link. If *traceroute* packets are generated keeping these addresses as the destination address, the adjacent nodes will send back *Destination Unreachable ICMP* packets to the source. This way, the IPv6 addresses of these routers can be known.

As real *traceroute* packets were not generated, the algorithm determined the number of interfaces that can be discovered this way and calculated the number of probing pairs these newly found interfaces would generate.

3.9 Summary

In this chapter, a probing space reduction algorithm based on SPF algorithm and *source routing* has been presented. The OSPFv3 based SPF algorithm and the *source routing* based algorithm were implemented together to reduce the redundant probing of *source routing*. OSPFv3 based algorithm discovers the full router level connectivity of the IPv6 network. The connectivity information along with the information on shortest paths are used to provide an efficient probing mechanism for *source routing*. A detailed discussion on both of these algorithms has been presented. This chapter also described how the probing space reduction algorithm can be used for network management and troubleshooting. Section 3.2 described the advantage of the algorithm in terms of discovering the full IPv6 address of the router interfaces. In the *source routing* based algorithm, probing by traces ensures probing all the interfaces (primary and secondary traces) that falls inside the region of primary traces. Moreover, the traditional approach probes the maximum number of interfaces possible. Therefore, the algorithm will discover the maximum number of full IPv6 addresses that can be discovered by *source routing*. This algorithm can be applied for other applications of *source routing* in a timely manner, provided that the network is running OSPFv3 protocol.

Chapter 4

Performance Analysis of Topology Discovery Algorithm and Probing Space Reduction Algorithm

In this chapter, the performance of both the topology discovery algorithm based on OSPFv3 and the probing space reduction algorithm are evaluated. The design and setup scenario were described in Chapter 3 elaborately. In this chapter, the performance and the output of the OSPFv3 based discovery is presented at first. Section 4.1 also presents the types of information that are discovered from the network along with the information used for the probing space reduction algorithm. Section 4.2 presents a detailed discussion on the performance of the probing space reduction algorithm which is closely related to the network size and the initial seeds placement in the network. Therefore, the algorithm has been analyzed for different network sizes with different number of initial seeds placed in different locations of the network.

4.1 Analysis of OSPFv3 based Topology Discovery Algorithm

To analyze the performance of the OSPFv3 based discovery algorithm, a set of OSPFv3 test networks has been configured in the emulator. Once the adjacency between the Quagga router and one of the routers of the emulated network is established, the routers start sharing live packets. The client side that is inside of Quagga, as was shown in Figure 3.3, captures these packets and sends them to the server side to be analyzed for topology discovery. The OSPFv3 based algorithm discovers the layer 3 connectivity between the routers along with their router id, interface id, prefix information and the associated cost of the links. One of the observations of the algorithm was that OSPFv3 discovered the full network topology. This means that all the routers, links and prefixes were discovered. Thus, the complete connectivity information of the network was achieved with this algorithm. To test the performance, networks with different numbers of routers (50, 60 and 70) were configured. The number of routers was selected based on the typical number of routers in one OSPF area. The number of routers in one area depends on several factors like CPU (Central Processing Unit) power, type of media, type of area, the number of LSAs, etc. [44]. Moreover, if the network is a mesh topology, the optimal size might reduce even more. The router LSAs must be kept under the IP MTU size. Moreover, a larger network would generate more router LSAs which are needed to be processed by all the routers. Each router maintains its LSDB and the routing table based on the LSAs. Thus, a scalable design also considers the CPU power of the routers. The document that CISCO publishes suggests fifty or fewer routers is the most optimal design for one OSPF area [44], although one area can be configured with more than one hundred routers. For this reason, all the networks in this research have been configured with at least 50 routers for performance analysis. The performance has been analyzed for 12 networks (OSPFv3) with different numbers of routers. Out of these 12 networks, 4 networks have 50 routers, as suggested by CISCO for one area [44], 4 networks have 60 routers and 4 networks have 70 routers. Table 4.1 provides the details of these networks with the average number of links per router, number of links in the networks, and the range of link cost selected for all the links.

Table 4.1 Characteristics of Test Networks

Network Size	No.	Average number of links per router	Number of links	Average number of links for a particular network size	Cost of links
50 Routers	1	2.92	73	82.5	1-10
	2	3.48	87		1-10
	3	3.76	94		1-10
	4	3.04	76		1-10
60 Routers	5	3.4	102	98.5	1-10
	6	3.62	108		1-10
	7	3.03	91		1-10
	8	3.1	93		1-10
70 Routers	9	3.11	109	114.5	1-10
	10	3.09	108		1-10
	11	3.27	115		1-10
	12	3.6	126		1-10

All the networks were manually configured, while making sure the network is always connected and looks as random as possible. The number of links per router and cost of links were configured without any previous assumption or bias. The number of links for each router was varied from 1 to 6. All the link costs are varied from 1 to 10. In this research, the link costs were configured in a symmetric manner. That means the two interfaces of a link were configured with the same cost. However, in practice, costs might be configured in asymmetric way. Thus, two interfaces of a link might have difference cost. The total number of links for all the test networks of a particular size is very close to each other. The average number of links is larger for networks with larger number of nodes. Networks with the same number of routers might have different number of links because of the different topologies.

The output of the OSPFv3 based algorithm was generated in terms of the connectivity between the routers. Each router was identified by their router id. Each interface of a router was identified by the interface id. To show an example on how the OSPFv3 based algorithm discovers the topology, a small network with 20 routers and 35 links has been configured. Figure 4.1 is drawn based on the output generated by the algorithm for a network of 20 routers.

Figure 4.1 shows that each router is identified by its router id. For example the router adjacent to the Quagga router has a router id 5.5.5.5. Each interface of the routers is identified by its interface id. As can be seen from the figure, router 5.5.5.5 has 4 interfaces with interface id 11, 22, 80, and 58. The bold numbers between the nodes are the link costs. The prefix information discovered by the algorithm is also shown between the links in the figure. For example, router 5.5.5.5 (interface id 80) is connected to router 10.0.87.6 (interface id 82) with link cost 3 and IPv6 prefix 2001:2::/64. The OSPFv3 based algorithm processes all the four types of LSAs (router LSA, network LSA, intra-area prefix LSA and link LSA) to build the connectivity of the network.

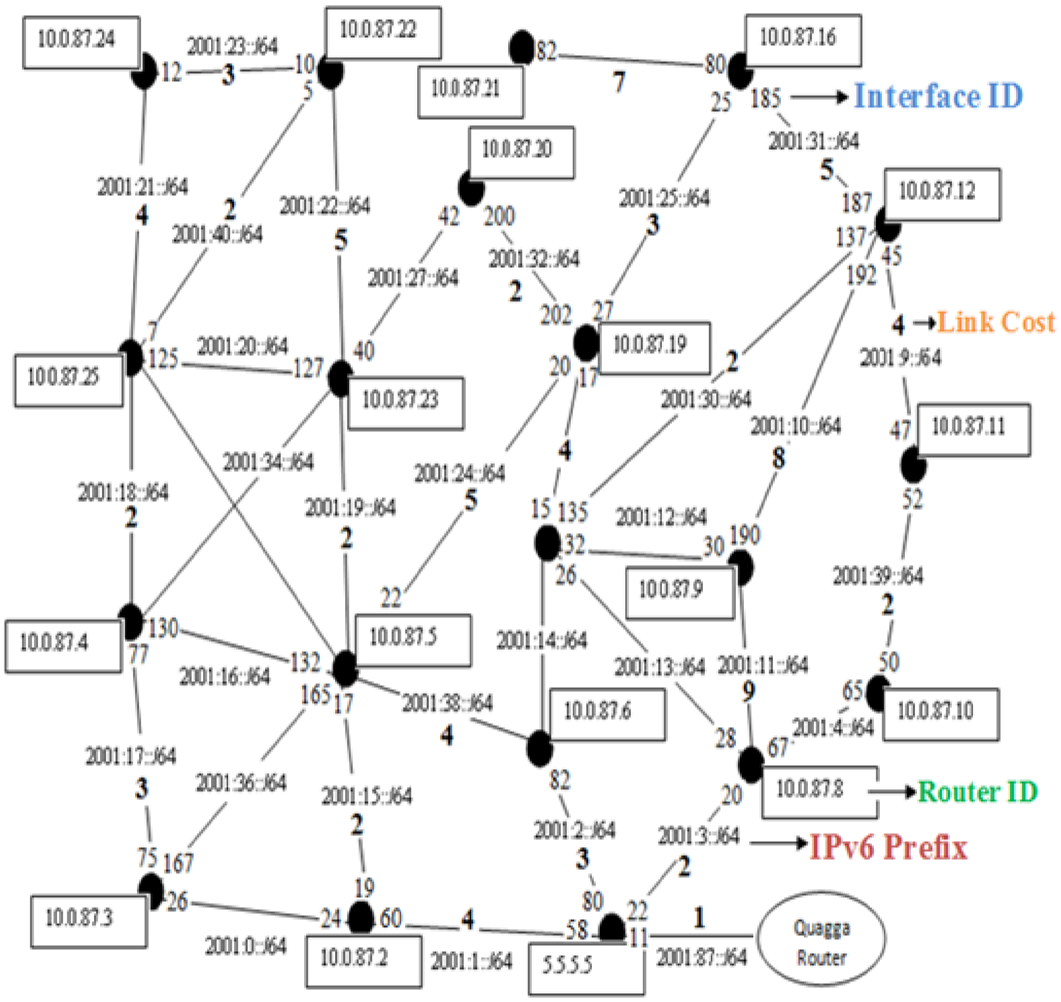


Figure 4.1 Network Topology Discovered by OSPFv3 Based Algorithm

Figure 4.1 is the full topology discovered by the OSPFv3 based algorithm. As can be seen, the full connectivity between the routers has been discovered. The discovered topologies for all the test networks (Table 4.1) were exactly the same as the emulated networks. Therefore, if the network (OSPFv3) is stable and if the Quagga router maintains a full adjacency with one of the routers of the network, the OSPFv3 based algorithm can discover the full router level topology of the network. On the other hand, as shown in Figure 4.1, only the subnet or prefix information of each link was discovered using OSPFv3. The full IPv6 address of the router interfaces, including both the prefix and the host portions, cannot be discovered by the OSPF based discovery alone. For example, the link between the routers 5.5.5.5 and 10.0.87.6 has the IPv6 prefix 2001:2::/64. This is only the 64-bit prefix of the full IPv6 address. The 64-bit host portion of the IPv6 address could not be discovered by the algorithm. Figure 3.2 of Chapter 3 showed the phenomenon of incomplete discovery of IPv6 addresses. This is the reason why we need another method to learn the host portion of the IPv6 addresses.

For all the OSPFv3 test networks, the discovery algorithm only took a few seconds to discover the topology. All the LSAs are flooded as soon as the Quagga router create adjacency with another router. The algorithm takes only a few seconds to analyze these LSAs and to build the topology. The discovery time for all the networks is almost the same as the LSAs are flooded as soon as the adjacency is established. Moreover, subsequent changes in the network (e.g. link cost) are flooded throughout the network and will be detected by the algorithm as soon as the changes happen.

4.2 Analysis of Probing Space Reduction Algorithm

OSPFv3 is based on Dijkstra's SPF algorithm. As the connectivity information of all the links along with their costs have been discovered, it is possible to find the shortest path from each node to every other nodes of the network. The probing space reduction algorithm builds upon this idea to find the paths that *source routing* packets would traverse through. The probing space is actually the number of probing pairs. In this chapter, these two terms are used interchangeably. A probing pair is defined as the pair of an intermediate node and a destination node.

Based on the concept and supported by experimental results, the number of *source routing* packets has been reduced based on these path information. Although, real *source routing* packets are not generated, this research focuses on how much probing pair reduction is possible if real *source routing* packets were generated.

The performance of the probing pair reduction has been analyzed for the 12 networks running OSPFv3 protocol with different numbers of routers as described in Table 4.1. The performance was then measured and compared for these networks in terms of probing space, number of links that were not probed by traces and number of links that had to be probed by the traditional approach.

4.2.1 Probing Space versus Number of Initial Seeds

In this section, the impact of the number of initial seeds on the probing space is presented. The number of initial seeds has been varied for different topologies. As discussed in Section 3.7.2, we form the primary traces based on the results of the *traceroute* to the given initial seeds. It means that each initial seed crates one primary trace. Further traces (i.e. secondary traces) are created from the routers found from the primary traces. Each new trace is formed between two known traces. Thus, the algorithm requires at least two initial seeds to start with. The number of initial seeds can vary from two to any number (up to the number of router in the topology). The algorithm will run with any router in the network as an initial seed. However, the farthest routers from quagga will perform as better seeds, as they will cover more nodes to be probed by traces. In this research, the focus on the seed selection is to choose the seeds as far as possible from quagga.

Initial seeds are chosen in such a way that they are distributed throughout the network. The whole region of the network is divided by the number of initial seeds. If the number of initial seeds is *n*, the whole region will be divided into *n* sections. Figure 4.2 shows an example of the seed selection method with 8 initial seeds (*n* = 8). The whole network was divided into eight sections. We selected one seed from the border of each section. If there are multiple nodes in the border of a section, one of them is selected. Each of these seeds will generate one primary trace. The results of the primary traces will be used to generate secondary traces later.

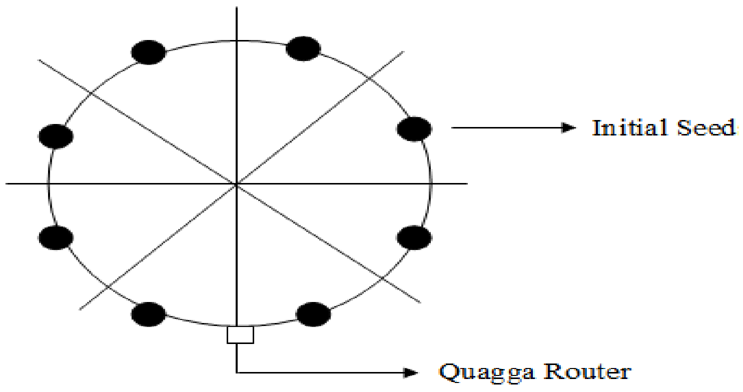


Figure 4.2 Example of the Seed Selection Method with 8 Initial Seeds

For illustration, Figure 4.3 shows the number of generated probing pairs (probing space) for one of the networks of 70 routers (Network 10 in Table 4.1), while the number of initial seeds is varied from 2 to 8. The results of all other networks are presented in the subsequent figures.

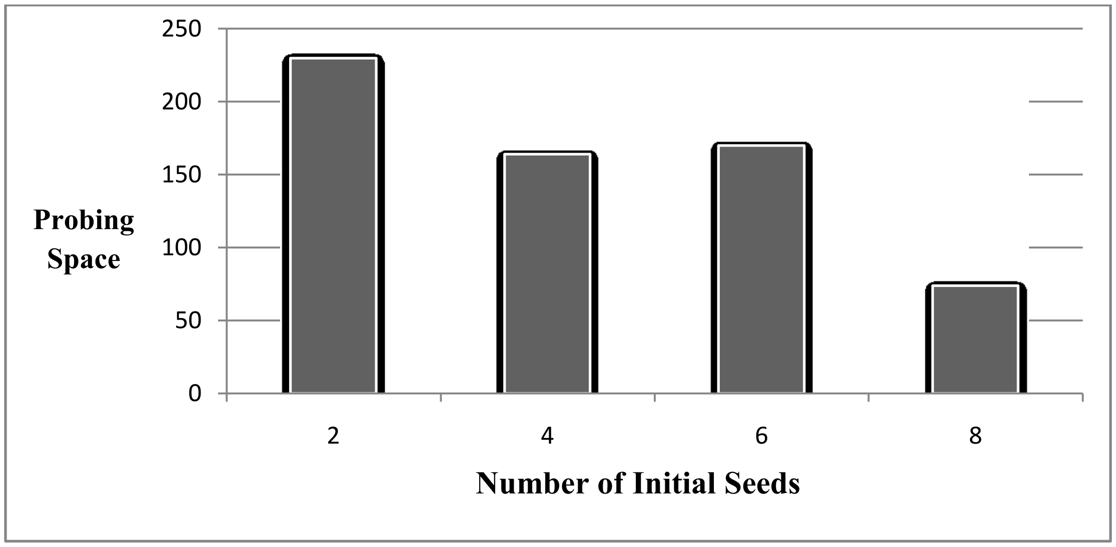


Figure 4.3 Probing Space vs. Initial Seeds for a Network of 70 Routers

Figure 4.3 shows that, in general, as the number of initial seeds is increased, the probing space is decreased. The probing space reduction algorithm can be divided into two steps as discussed in Chapter 3:

1. Probing by forming traces (primary and secondary) using the initial set of seeds.
2. Probing by the traditional approach for uncovered region after step 1

A larger number of initial seeds generally not only produces more primary traces than a smaller number of seeds, but also produces a larger number of secondary traces based on the result of the primary traces. The number of overall traces is also larger for larger number of initial seeds. This is because all the traces are formed between the primary traces. If the primary traces cover more region of the topology, more secondary traces will form between these primary traces. As the number of primary traces equals the number of initial seeds, more initial seeds will cover more region of the network which in turn results in the creation of more traces. As discussed in Section 3.7.2, the links that cannot be probed by forming traces will be probed by using the traditional approach of *source routing*. For instance, if there are 10 nodes that cannot be probed by forming traces, then there will be a probing space of 10(10 - 1) = 90 since the probing space for n nodes is $n(n\text{-}1)$ as discussed in Chapter 2. If the number of initial seeds is smaller, the number of links that needs to be probed by the traditional approach gets higher. This is the primary reason of a larger probing space when the number of initial seeds is lower. This implies that if there are more links that are not probed by forming traces, the overall probing space will be larger. Therefore, more initial seeds ensure greater coverage and also indicate formation of more secondary traces and less links to be probed using the traditional approach.

However, increasing the number of seeds does not always reduce the probing space. In Figure 4.3, the probing space for 6 seeds is higher than the probing space for 4 seeds. This is because the *source routing* packets form the traces based on the shortest paths. A shortest path will not necessarily reveal more links to probe. It might traverse through a path that has already been probed before. As a result, this path will not help creating any new traces. If no new trace is created, the probing space will not decrease as the probing space decreases only if there are more traces. This means that the location of the seeds is also an important parameter to consider.

Figure 4.4 shows the average probing space of all the 12 test networks based on their size.

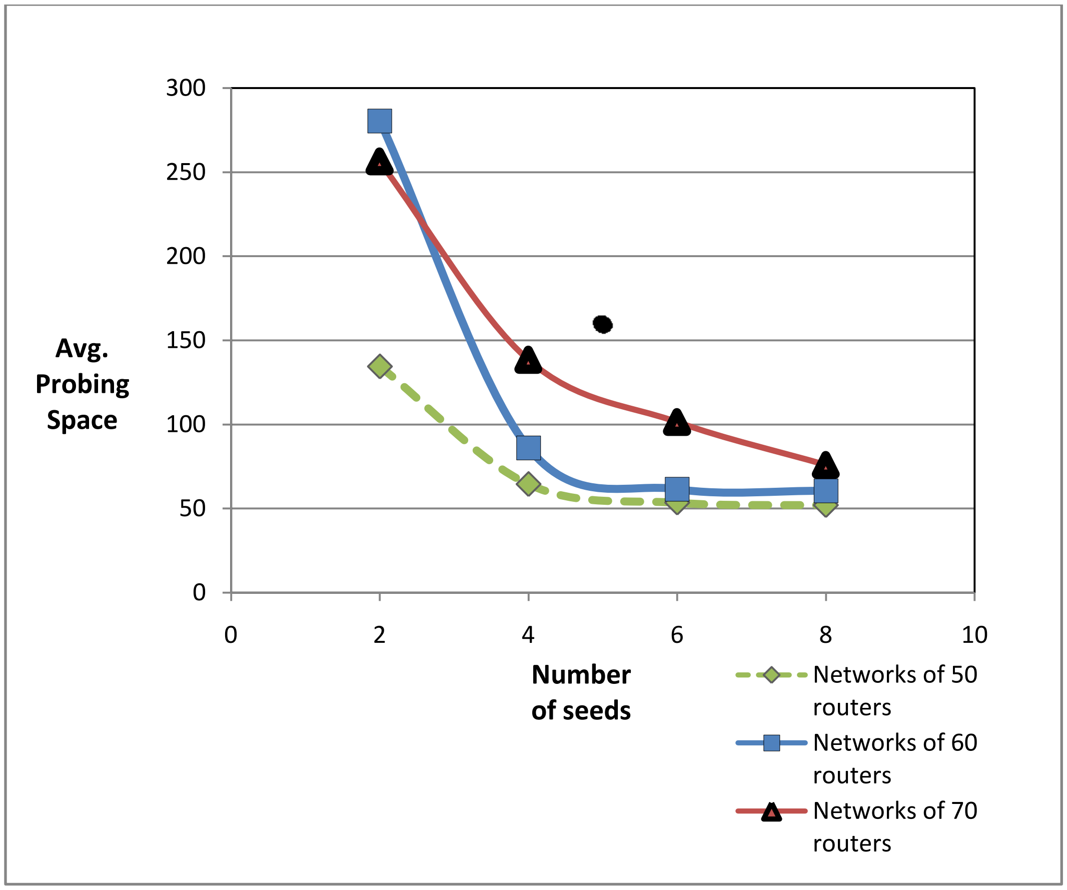


Figure 4.4 Average Probing Space for Different Network Size

Figure 4.4 shows how the probing space varies for different network sizes and different number of initial seeds. The results of probing space for all the network sizes show a decreasing trend, as the number of seeds increases. Specifically, some observations have been made based on the experiments:

1. A general observation is that larger networks generate more probing pairs. For example, when the number of seeds is 4, the networks with 50 routers generated about 60 probing pairs on average, whereas the networks with 60 and 70 routers generated more probing pairs. As discussed earlier, if more links are needed to be probed by the traditional approach, the probing space becomes larger. Otherwise, the probing space generated by the traces does not vary a lot. The number of links needed to be probed by the traditional approach becomes higher, if for a fixed number of

seeds the network size is bigger. Therefore, more initial seeds are required for a network with more routers and links.

A difference on the above discussion happened when the number of seeds was 2. As can be seen in Figure 4.4, the average probing space for 60 routers is higher than the average probing space for 70 routers. As OSPFv3 uses the shortest path algorithm, some primary traces might take a path in such a way that does not create many secondary traces, resulting in larger probing space generated by traditional approach. One test network of 60 routers generated 720 probing pairs which made the average number higher than that of 70 routers.

2. Another observation is that the trend of probing space becomes flat after a certain number of seeds. In the figure, when the number of seeds is more than 4, networks with 50 routers and 60 routers, start generating almost the same number of probing pairs. Thus, the algorithm only needs a certain number of seeds to form traces throughout the network.

3. Another observation is that the confidence intervals (Figure 4.5, 4.6, 4.7) for probing space decreases as the number of initial seeds increases. The average probing space by traditional approach varies a lot for a small number of seeds. Thus the deviation of the average probing space is larger. The confidence intervals for larger number of initial seeds is relatively small. Confidence intervals of probing space for 95% confidence level for all the 12 network topologies are shown in Figures 4.5, 4.6, and 4.7. The figures shows the highest confidence interval, when the number of seeds is 2.

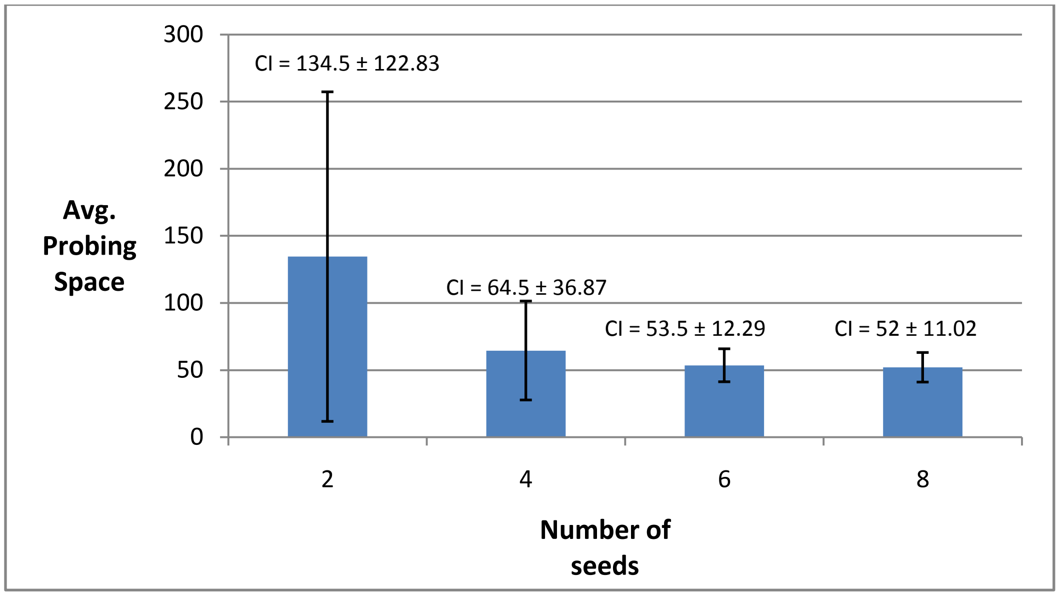


Figure 4.5 Confidence Interval of Probing Space for Networks of 50 Routers

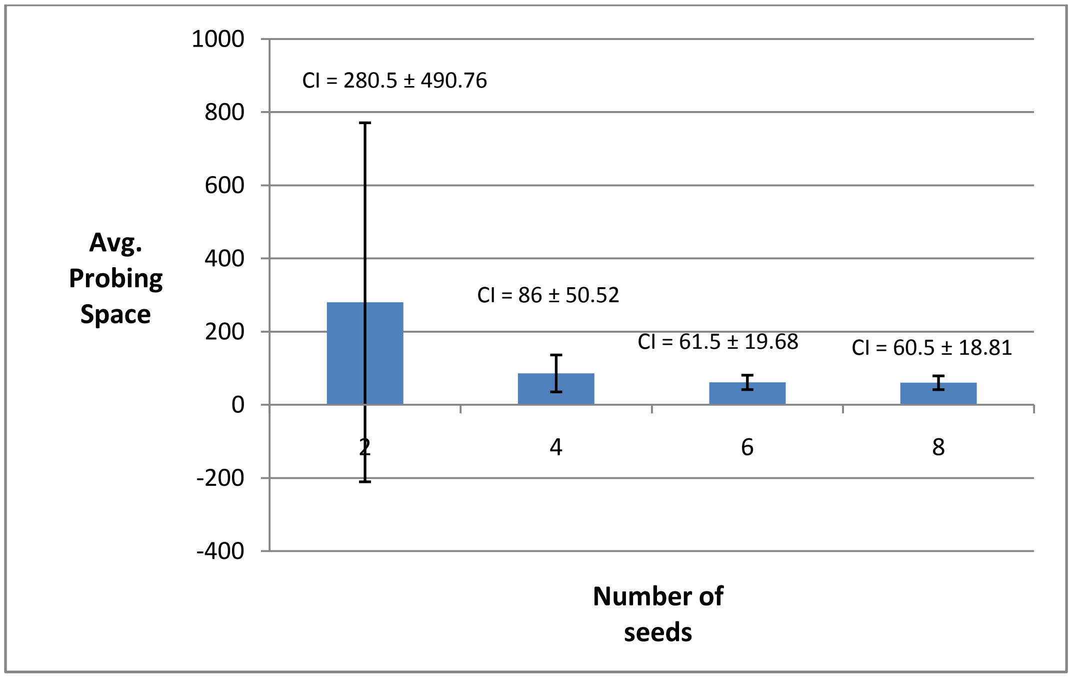


Figure 4.6 Confidence Interval of Probing Space for Networks of 60 Routers

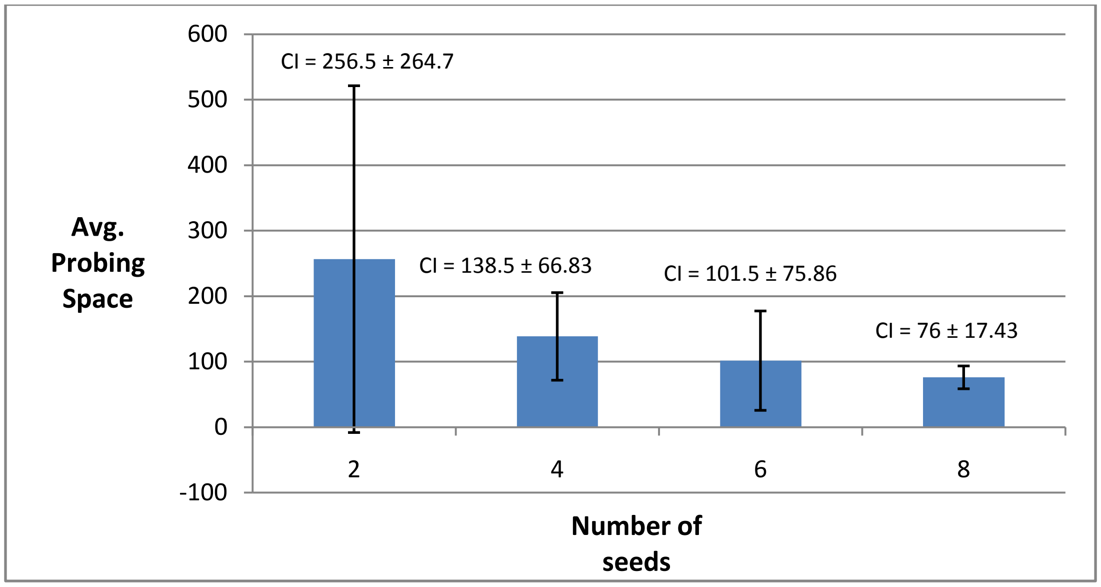
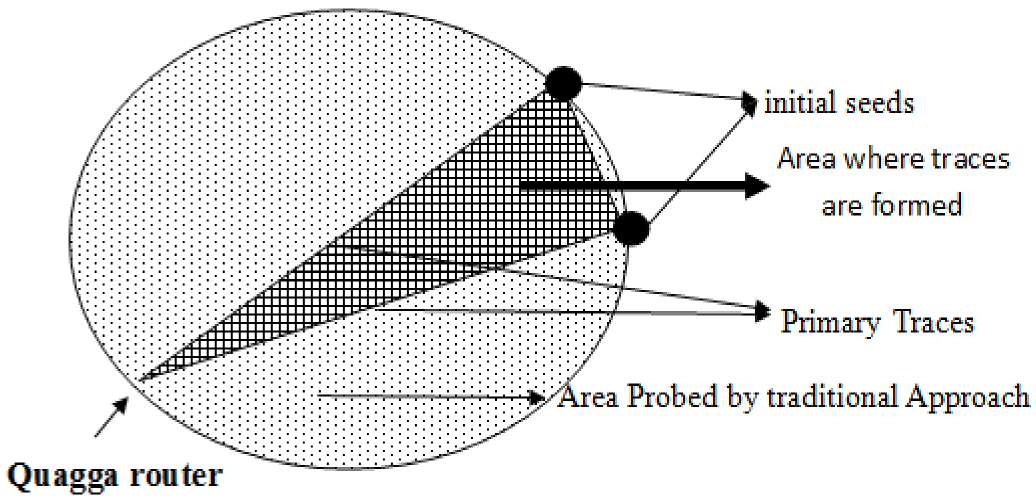


Figure 4.7 Confidence Interval of Probing Space for Networks of 70 Routers

The deviation of probing space generated by traditional approach is very large when the number of seed is 2. Therefore, the confidence intervals for 2 seeds is the largest for all the topologies

4.2.2 Probing Space versus Location of Initial Seeds

As discussed in Section 4.2.1, if the primary traces cover more regions of a network, more traces will be generated. This reduces the probing space of *source routing*. The source routing based algorithm will work with any router as a possible seed. However, the furthest routers from quagga will perform as better seeds, as they will cover more region for primary and secondary traces to be formed. In this research the seeds were chosen in such way that they are at the furthest possible locations from quagga. Figure 4.8 depicts how the location of the initial seeds influences the calculation of the probing space. This figure can also be described in terms of number of seeds from the previous section.



initial seeds
Area where traces are formed
Primary Traces
Area Probed by traditional Approach
Quagga router

Figure 4.8: Influence of Seed Location on Probing Space

Figure 4.8 shows a scenario for two seeds that are closely located. Two primary traces are first formed to these seeds from the Quagga router. All the subsequent traces (i.e. secondary traces) will be formed between these primary traces. This is because all the intermediate routers find the shortest path that gives the most number of links (not probed yet) to the routers of other traces. The routers and links that falls outside the primary traces (formed by initial seeds) will have to be probed with the traditional approach. Therefore, closely located seeds will generate more probing space. In Figure 4.9, the impact of the number of hops between the initial seeds is presented. In this study, 6 seeds were selected at different locations in the 4 OSPF networks containing 70 routers (networks 9, 10, 11, 12, as shown in Table 4.1). These seeds are selected from the edges of the OSPF networks. The seeds were selected based on the number of hops between each pair of routers. For example, the distance between these 6 seeds might be one hop, two hops, three hops, and so on. We varied the distance between these seeds starting from one hope to five hops. Thus, the seeds are most closely located when they are one hop away from each other.

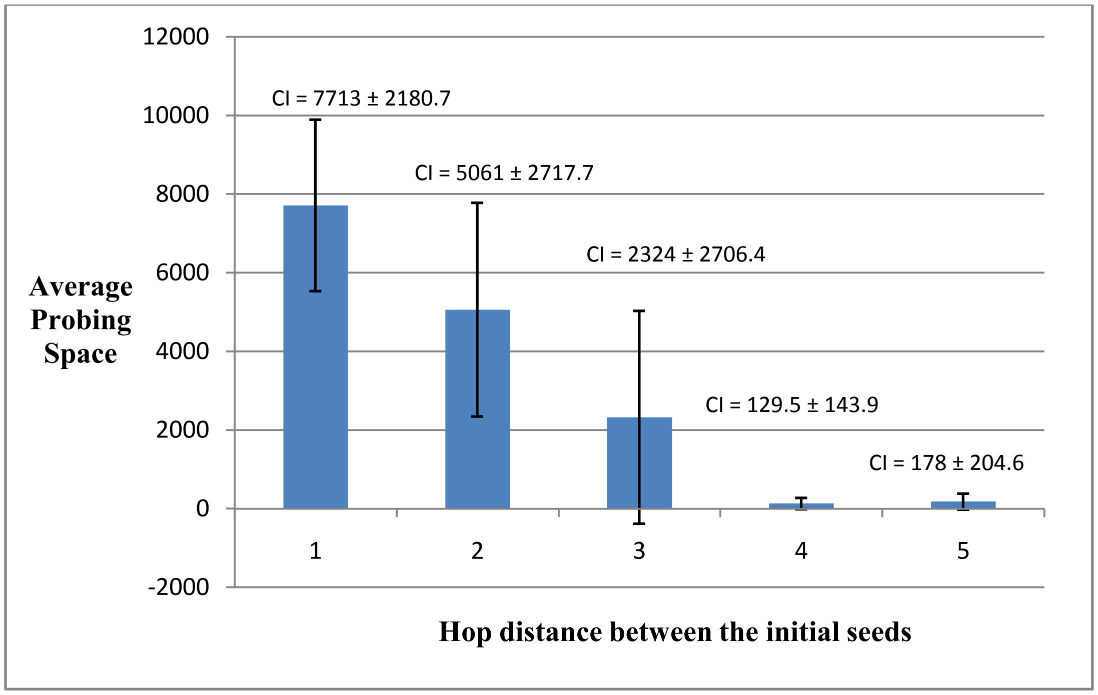


Figure 4.9 Average Probing Space vs. Seed Location

As the hop count between each two initial seeds is increased, the probing space decreases. A huge number of *source routing* packets (close to 8,000) is generated when the seeds are 1 hop away. The probing space becomes flat when the hop distance is more than 3. Moreover, the 95% confidence intervals are also very large when the hop count is less than 4. The probing space by the traditional approach varies a lot for a small number of seeds. Thus the deviation in overall probing space is very large. However, the confidence intervals become smaller when the hop count between the seeds is higher.

4.2.3 Probing Space by Traces versus by the Traditional Approach

As discussed in Chapter 3, the probing space reduction algorithm has two parts: probing by traces (primary and secondary) and probing by traditional approach. Probing by traces is responsible for reducing the average number of probing pairs. As the overall probing space decreases for a larger number of seeds, the probing space generated by the primary traces increases. Figures 4.10, 4.11, and 4.12 show the results for all the test networks. As the number of seeds increases, the primary traces cover more regions. Because of this, more secondary traces are formed and thus decreasing the probing space by the traditional approach.

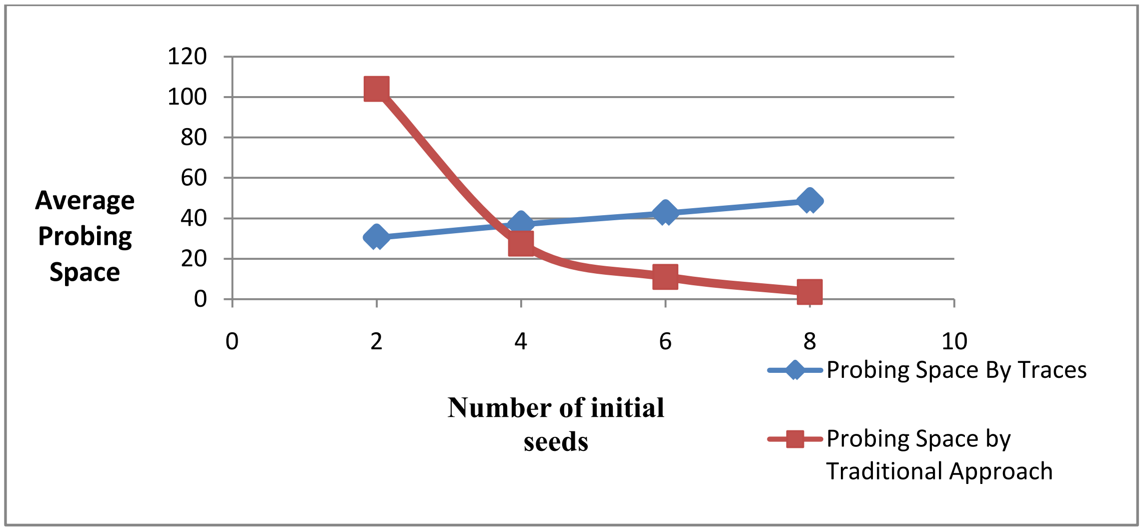


Figure 4.10 Average Probing Space by Traces vs. by Traditional Approach for Networks of 50 Routers

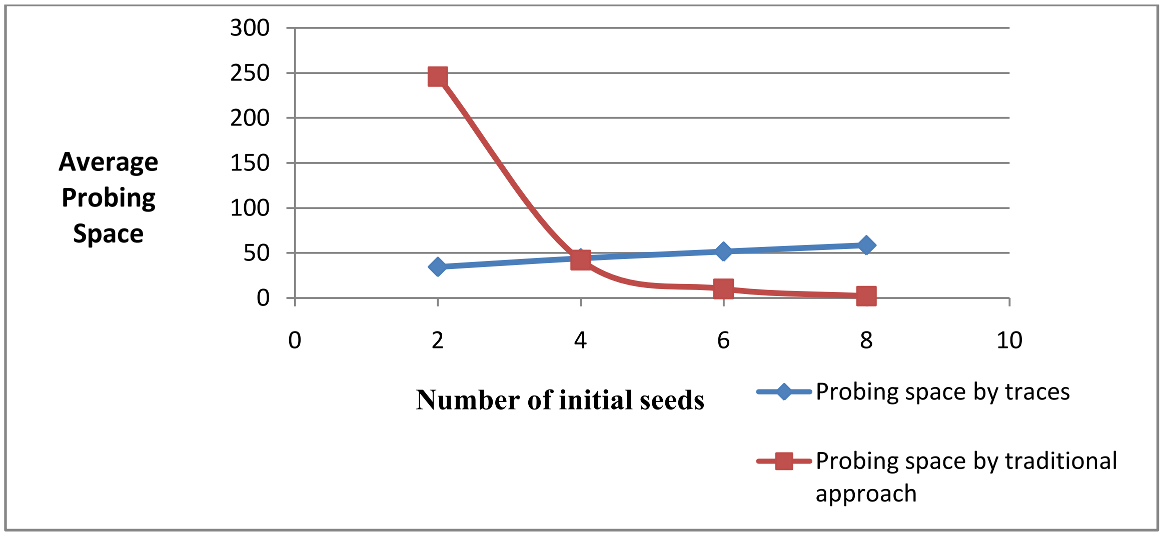


Figure 4.11 Average Probing Space by Traces vs. by Traditional Approach for Networks of 60 Routers

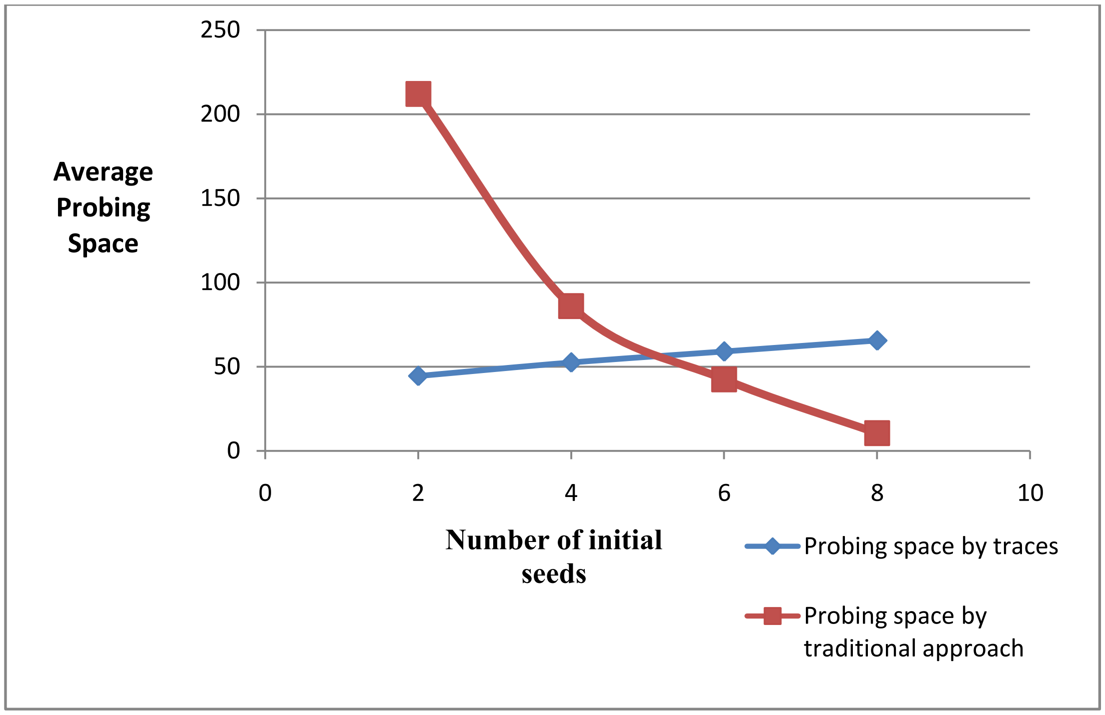
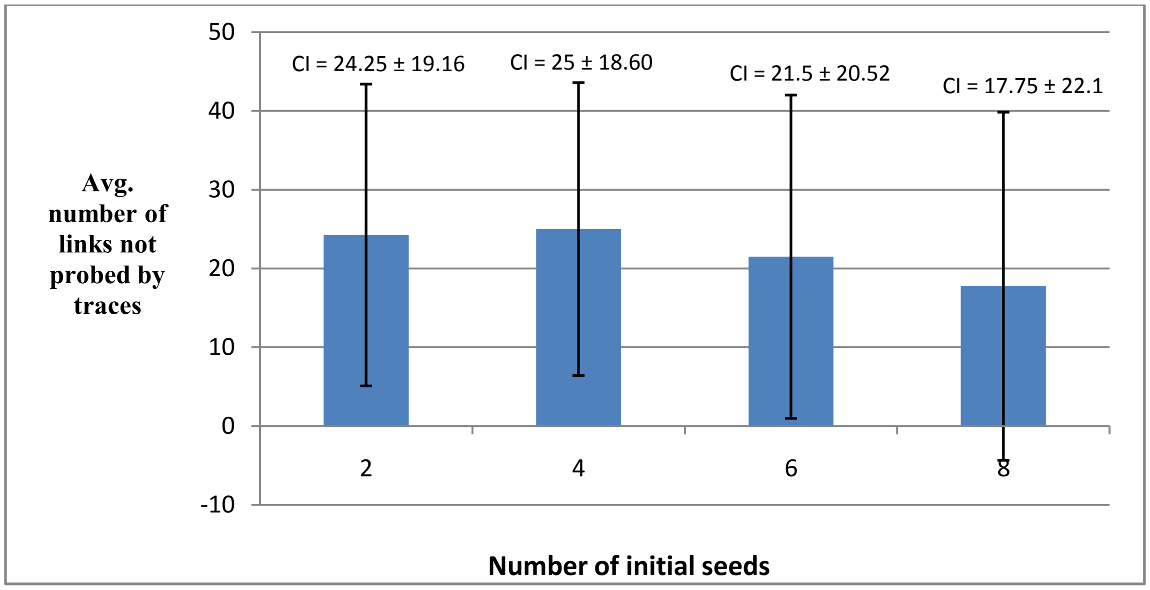


Figure 4.12 Average Probing Space by Traces vs. by Traditional Approach for Networks of 70 Routers

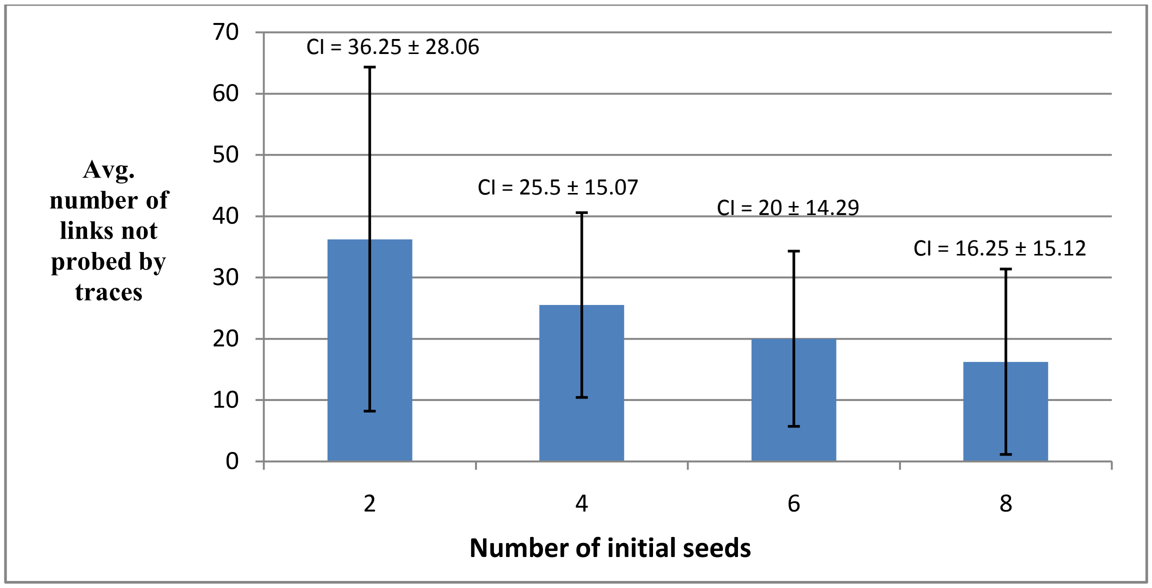
4.2.4 Probing Space in Terms of Links Not Probed by Traces

The links that are not probed by the traces (primary and secondary) will be probed using the traditional approach. Links that are not probed by traces are the links that fall outside the shaded region in Figure 4.8. These links are not in between the primary traces. The traditional approach ensures the maximum number of links to be probed out of these traces. As discussed in the previous sections, the probing space increases rapidly if there are more links to be probed using the traditional approach. Figures 4.13, 4.14, and 4.15 show the average number of links that could not be probed by traces for four networks with 50, 60 and 70 routers, respectively. These are the links that need to be probed using the traditional approach. As we can see, all the figures shows a decreasing trend as the number of initial seeds increases. As the number of seeds increases, more primary and secondary traces are formed thus decreasing the number of links not probed by traces.

The 95% confidence intervals are shown by the vertical lines. The intervals for the networks with 60 and 70 (Figures 4.14, 4.15) routers showed a decreasing trend. This is because the deviation of the number of links (not probed by traces) is higher for a small number of initial seeds. However, the confidence intervals for 8 seeds are larger than the intervals for 6 seeds in Figures 4.14 and 4.15. This is because for some networks with 60 and 70 routers, 8 initial seeds covered the whole region and made the number of links not probed by traces very small. The same phenomenon happened with the networks of 50 routers (as shown in Figure 4.13), when the number of initial seeds were 6 and 8.



4.13 Average Number of Links not Probed by Traces for Networks of 50 routers



4.14 Average Number of Links not Probed by Traces for Networks of 60 routers

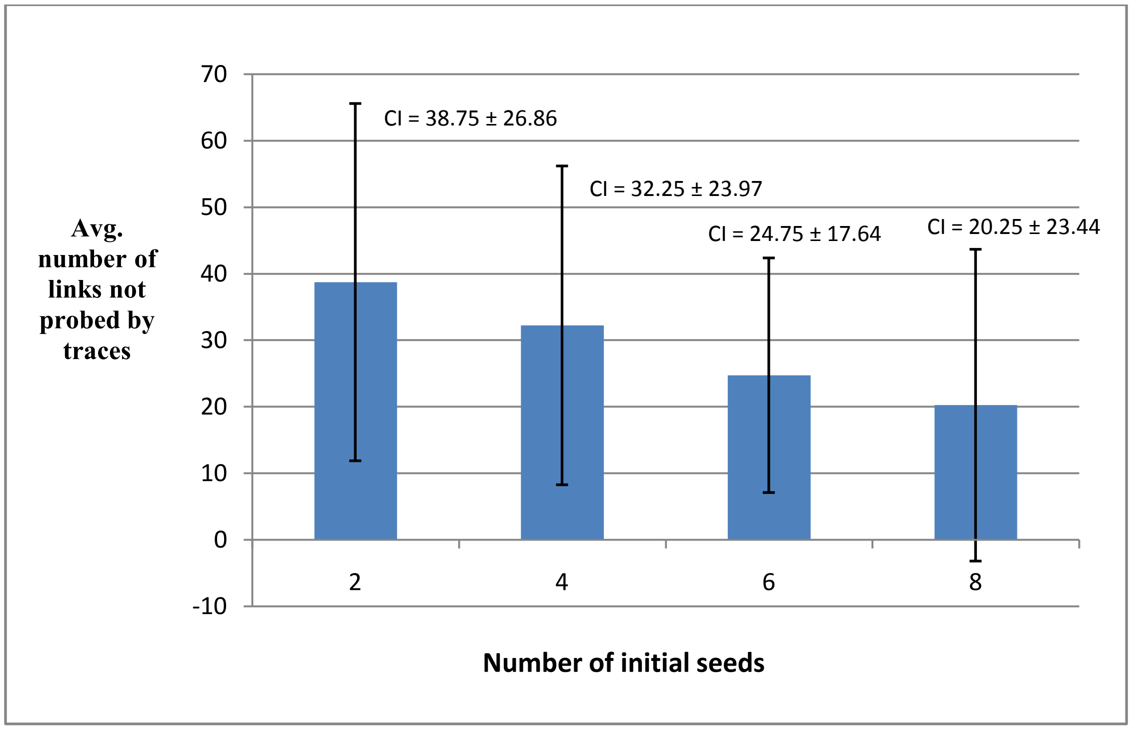


Figure 4.15 Average Number of Links not Probed by Traces for Networks of 70 routers

4.2.5 Performance Analysis with Manhattan Grid Topology

To get a clearer picture of the performance of the probing space reduction algorithm from the topology perspective, we executed the algorithm on a Manhattan topology for a basic grid of 6 × 6 nodes. As the Manhattan topology has an equal dimension in both axes, keeping seeds on the edges of the grid will give a clearer picture of how the reduction algorithm works in terms of the number of seeds than selecting seeds in the middle. Figure 4.16 depicts the probing space and Figure 4.17 illustrates the number of links that could not be probed by traces. The number of initial seeds is varied from 2 to 4 located in the corners of the grid.

As demonstrated in Figure 4.16, the probing space decreases sharply from 2 seeds to 3 seeds. From Figure 4.17, it can be seen that, even if the number of seeds is 4, located in the four corners of the grid, there are about 14 links that could not be probed by the traces. This is because the primary traces were not formed exactly through the edges of the grid. There will be a few links and routers that falls out of the region where the traces are built.

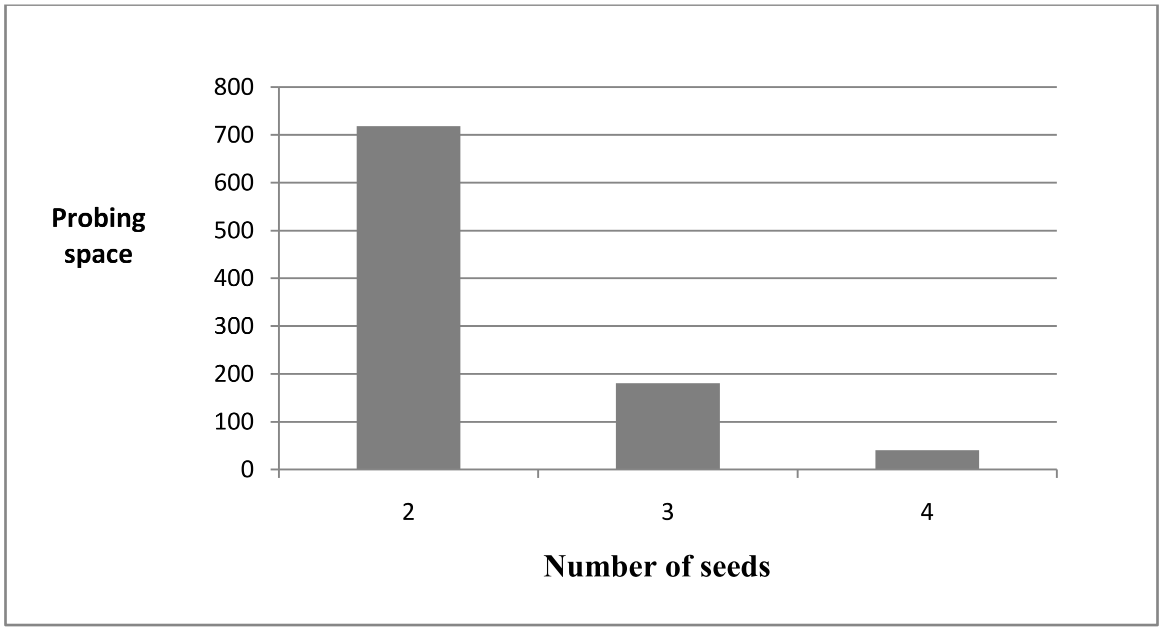


Figure 4.16 Probing Space for a 6 × 6 Manhattan Topology

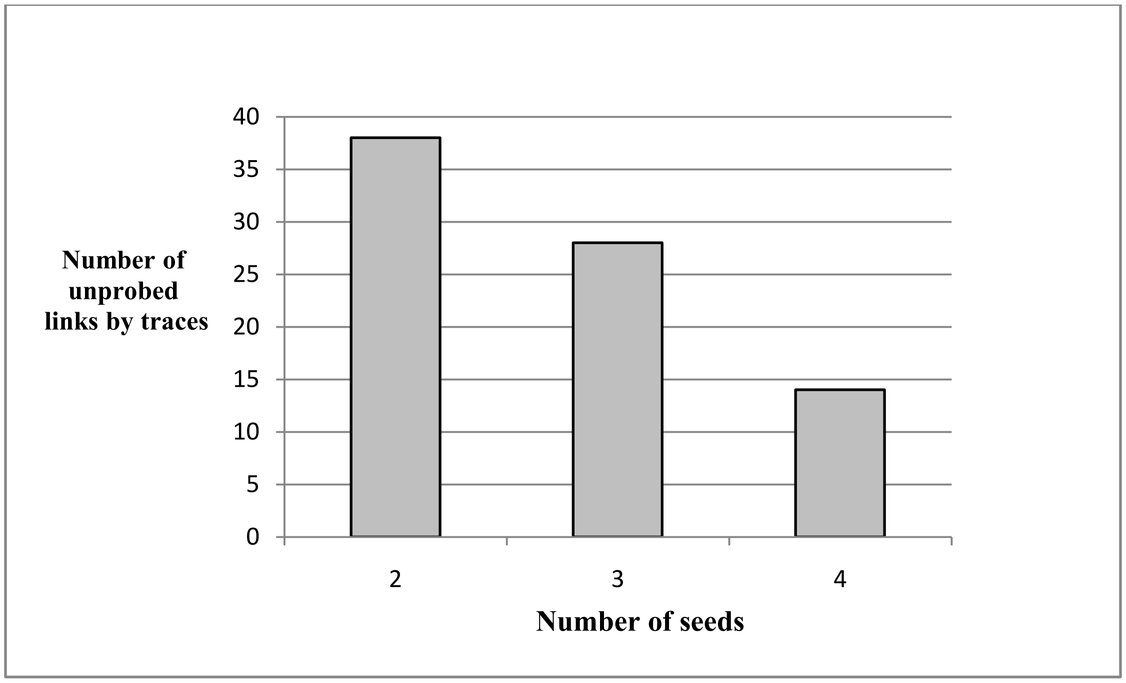


Figure 4.17 Number of Links not Probed by Traces for a 6 × 6 Manhattan Topology

4.3 Performance Analysis in Terms of Related Works

As discussed in Chapter 2, not much research has been conducted on the reduction of probing space. The authors in [3], [4] and [5] implemented *source routing* to keep the redundant probing as low as possible. ATLAS [3] had a total number of probing paths of 308,887 for 2,420 routers which is a huge number. This huge number of probing paths took eight weeks for ATLAS to probe the whole 6Bone topology. ATLAS used caching and parallelism techniques to increase their probe engine performance to reduce redundant probing. ATLAS considered the whole administrative domain of 6Bone. The number of initial seeds for ATLAS was 426. The research proposed in this thesis focused on only the area level reduction of probing space. The objective of the thesis is only for probing space reduction for one OSPF area. Therefore, we cannot directly compare our results with ATLAS. The authors in [5] also introduced an improved solution of *source routing* mechanism. From a network of 1,000 nodes and 200 initial seeds, they pruned off a total number of 410,376 pairs. The overall probing space of their approach was 203,496. Both the above papers have reduced the probing space. In a comparative measurement, the probing space reduction algorithm of this research showed an improvement into two categories:

- Reduction of the probing space
- Reduction of the initial number of seeds

As shown in Figure 4.4, the average number of probing pairs for a network of 70 routers with 4 initial seeds is only about138. The reduction was significant because the algorithm knows the OSPF topology. The algorithm discovers the OSPF topology first and then uses the topology information to generate the minimum number of probing pairs. As the connectivity information was known, the reduction of the probing space was higher. Although, this research focuses on area level probing space reduction, an approximate comparison between the proposed approach and other approaches is shown in Table 4.2.

Table 4.2 Comparison of Probing Space Reduction Algorithms

Number of routers in one area	Probing space of ATLAS [3] for 2420 nodes	Estimated probing space of 2420 nodes for x nodes/Area	Probing space of [5] for 1000 nodes	Estimated probing space of 1000 nodes for x nodes/Area
50	308,887	3122	203,496	1290
60	308,887	3469	203,496	1434
70	308,887	4789	203,496	1979

x = 50, 60, 70; where x is number of nodes per area

Table 4.2 is an approximate comparison of results between the probing space reduction algorithm and two other approaches. As this research focuses on only one OSPF area, the probing space was approximated for the number of routers for the other approaches. One advantage of combining the OSPF based and *source routing* based algorithm is that the algorithm will be able to probe multiple areas of an entire domain. This means that if the probing machine (machine where the topology discovery algorithm and *source routing* algorithm runs and generate packets) is connected to one router from all the areas of an entire domain, the algorithm can simultaneously discover the connectivity of all the areas as well as probe to all the areas to reduce probing space. Both the OSPFv3 based algorithm and *source routing* based algorithm can be performed separately on all the areas. Thus, the proposed algorithm is linearly approximated. For example, as depicted in Table 4.2, a network with 1000 nodes (like [5] has) will require 20 areas of 50 nodes. The average probing space for an area of 50 nodes is 64.5, as shown in Figure 4.4. Hence, the estimated probing space for 20 areas is $64.5 \times 20 = 1290$. Table 4.2 shows that the probing space reduction is significant with the proposed algorithm.

Another important observation is that the proposed algorithm started with the minimum number of seeds. For a topology of 70 routers, using 8 seeds (~11.4%) was proven to be sufficient. On the other hand, for a topology of 2,420 routers, ATLAS started with 426 seeds (~17.6%). The authors in [5] started with 200 initial seeds for a network of 1000 nodes (~20%). The number of initial seeds is small as the proposed algorithm discovers the topology before reducing the probing space with the initial seeds.

Although this algorithm is based on one OSPFv3 area, it can be easily implemented for an entire autonomous system that covers more than one area. The Quagga router interface can maintain adjacency with multiple OSPF areas. This way, the algorithm will work for all the OSPF areas of an autonomous system. Moreover, if multiple instances of OSPF are present, the algorithm will not face any problem as it will run on each instance separately.

4.4 Summary

In this chapter, we analyzed the performance of both the OSPFv3 based and *source routing* based algorithm. At first, the performance of the OSPFv3 based topology discovery algorithm was discussed. It was shown that the discovery algorithm can discover the full OSPFv3 area level topology. Although the discovery algorithm discovers the router id, interface id, link cost and prefix information of each link, it cannot discover the full IPv6 addresses of the routers. As a result, the OSPVv3 based algorithm needs to be used in conjunction with another method in order to discover the full topology. The source routing based algorithm ensures that the maximum number of interfaces is probed. Therefore, the discovery of the full IPv6 addresses will be maximum.

Next, the performance of the probing space reduction algorithm was discussed. It was found that the probing space decreases if the algorithm starts with more initial seeds. But after a certain value, even if the number of seeds is increased, the value of the probing space does not differ much. Thus, the algorithm needs only a certain amount of seeds to probe the maximum number of links of the network. The locations of the initial seeds also have a huge influence on the calculation of the probing space. Seeds with more number of hops between them (i.e. evenly distributed in the network) are likely to generate a smaller number of probing pairs. This chapter also demonstrates the comparison between two segments of the algorithm: probing space by traces and probing space by traditional approach. It was found that more traces are formed if the number of seeds is increased. For a small amount of initial seeds, probing space by traditional approach is larger than probing space by traces. The first part of the algorithm is actually responsible to reduce the probing space. Section 4.3 discussed the proposed algorithm in terms of related works on probing space reduction. As the proposed algorithm focuses on only one OSPFv3 area, a relative comparison was estimated.

Chapter 5

Conclusions and Future Work

In this chapter, an overview on the contribution of this thesis is presented. This chapter also presents the limitations of both the network topology discovery and probing space reduction algorithms. The final section concludes this thesis with possible future work in the related field.

The primary goal of this research was to develop a network management algorithm that would discover the IPv6 network topology with an efficient mechanism for troubleshooting possible network phenomenon. In this research, a topology discovery algorithm was developed that discovers an IPv6 network and reduces the number of probing pairs of *source routing* that facilitates efficient network management and troubleshooting.

The algorithm is an area discovery mechanism for networks running OSPFv3. Chapter 4 showed that the topology discovery algorithm discovered all the nodes and links of an OSPFv3 area with all the router ids, interface ids, prefix lists and the associated costs of the links. However, the discovery algorithm cannot discover the full IPv6 address of the links.

The probing space reduction algorithm is based on the information discovered by the OSPF discovery algorithm and it uses the Dijkstra's SPF algorithm to reduce the number of redundant probing of *source routing*. Experimental results presented in Chapter 4 demonstrated significant reduction in terms of probing space. For instance, to probe the maximum number of links of an OSPF network of 70 routers, the reduction algorithm needs an average probing space of only 138.5 with 4 initial seeds.

The algorithm showed improvement in two areas:

- Reduction of redundant probing and
- Reduction on the number of initial seeds needed for an efficient performance of the algorithm.

The probing space reduction is significant as the algorithm took the primary assumption that it already know the router level connectivity of the IPv6 network.

As the number of *source routing* packets is greatly reduced, frequent probing operations using this algorithm can be performed to give the network administrator a constant and live picture of the network. One of the main advantages of this algorithm is that it will not generate so much traffic in the network. As the number of *source routing* packets is reduced, it is assumable that the probing time will also be reduced. Thus, the algorithm can be performed in a frequent basis to detect any recent change or failure in the network.

5.1 Limitations

The main weakness of the algorithm is that it takes the primary assumption that it knows the connectivity and SPF information of the network. This the main reason that the probing space reduction was significant. It might not be always possible to discover the network topology. The *source routing* based algorithm relies on the fact that, it knows the SPF and connectivity information of the network.

Another weakness of the algorithm relies on the number of initial seeds chosen and the location of the initial seeds. If the number of seeds and the location of seeds are poorly selected, the algorithm may perform like the traditional approach of *source routing* if the seeds are clustered together. The number and the location of the initial seeds determine how many probing operations should be performed by the reduction approach and how many probing operations should be done by traditional approach. However, in practice, the location of those initial seeds is unlikely to be grouped in a small area. In Chapter 4, a detailed discussion on how the initial seeds should be selected for a better performance is presented.

Another limitation of the algorithm is the security issues of *source routing*. As discussed in Chapter 2, *source routing* causes security concerns in terms of Denial of Service attack (DoS). For this reason, a number of ISPs disable the *source routing* capability of their routers. This is a serious security concern in public networks. However, inside of campus networks or for small administrative domains, the administrators often enables *source routing* for topology discovery, network management and troubleshooting. In the gateway routers, the *source routing* is disabled so that, no *source routing* packet can come inside the campus network.

5.2 Future Work

Source routing is an important mechanism to discover full IPv6 addresses that cannot be discovered only by OSPF based discovery. However, *source routing* in IPv6 does not necessarily discover the full IPv6 address of all the router interfaces. The ICMP *Time Exceeded* packet might be originated from any of the interfaces of the routers. Thus *source routing* does not guarantee how many IPv6 addresses can be discovered. This might be a field of possible future work. A possible solution might be having multiple probe engines placed in different locations of the network. The reduction algorithm will be implemented in multiple probe engines, and all the probe engines will probe the nodes and links of the network from different location of the network. This means that the algorithm will run in more than one machine placed in different locations of the network.

The basic idea of the probing space algorithm might be extended to other types of routing protocols such as RIP, IS-IS etc., although the mechanism can be different for other routing protocols. Moreover, the algorithm works on the primary idea of shortest paths. Therefore, only the basic idea can be adopted for other protocols.

References

[1] I. Astic, and O. Festor, “A Hierarchical Topology Discovery Service for IPv6 Networks,” *Proc. of IEEE INFOCOM*, PP. 497-510, June 2002.

[2] Z. Shen, and G. Zhou, “Research of Topology Auto-discovery Method in IPv6 Access Network,” *Computer Engineering*, Vol. 32, No. 19, PP. 136-138, 2006.

[3] D. Waddington, F. Chang, R. Viswanathan, and B. Yao, “Topology Discovery for Public IPv6 Networks,”*ACM SIGCOMM Computer Communications Review*, Vol. 33, PP. 59-68, 2003.

[4] S. Dong, J. Li, L. Zhang, and J. Deng, “A Novel Algorithm of IPv6 Network Topology Discovery for Campus Network,” *Proc. of International Conference on Computer Science and Service System*(CSSS), PP. 68-71, 2011.

[5] Z. Mingming, and L. Junyong, “An Improved Solution for IPv6 Network Topology Discovery Based on Source Routing Mechanism,” *Proc. of International Conference on Communications Software and Networks*, PP. 279-282, 2009.

[6] X. Kou, and Q. Wang, “Discovering IPv6 Network Topology,” *Proc. of IEEE International Symposium on Communications and Information Technology*, PP. 1322-1325, 2005.

[7] T. Narten, E. Nordmark, and W. Simpson, “Neighbor Discovery for IP Version 6,” IETF RFC-2461, December 1998.

[8] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz, “Topology Discovery in Heterogeneous IP Networks,” *Proc. of IEEE INFOCOM*, PP. 265-274, 2000.

[9] B. Lowekamp, D. O’Hallaron, and T. Gross, “Topology Discovery for Large Ethernet Networks,” *Proc. of ACM SIGCOMM*, PP.237-248, 2001.

[10] R. Govinand, and H. Tangmunarunkit, “Heuristics for Internet Map Discovery,” *Proc. of INFOCOM*, PP. 1371-1380, 2000.

[11] H. Burch, and B. Cheswick. “Mapping the Internet,” *IEEE Computer*, Vol. 32, PP. 97-98, Apr. 1999.

[12] B. Huffaker, M. Fomenkov, D. Moore, and K. Claffy, “Macroscopic Analysis of Infrastructure: Measurement and Visualization of Internet Connectivity and Performance,” *Proc. of PAM*, PP. 23-24, 2001.

[13] R. Siamwalla, R. Sharma, and S. Keshav, “Discovering Internet Topology,” *IEEE INFOCOM*, PP. 1-16, 1999.

[14] G. Mansfield, M. Ouchi, K. Jayanthi, Y. Kimura, K. Ohta, and Y. Nemoto, “Techniques for Automated Network Map Generation Using SNMP,” *Proc. of INFOCOM*, 1996.

[15] B. Huffaker, D. Plummer, D. Moore, and k. Claffy, “Topology Discovery by Active Probing,” *Symposium of Applications and the Internet*, Japan, PP. 90-96, 2002.

[16] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, “Topology Inference in the Presence of Anonymous Routers,” *Proc. of IEEE INFOCOM*, 2003.

[17] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP Topologies with Rocketfuel," *Proc. of SIGCOMM*, PP. 19-23, August 2002.

[18] J. Reckard, "Mapping the Internet with Traceroute," *Boardwatch Magazine*, vol. 10, 1996.

[19] B. Donnet, T. Friedman, and M. Crovella, “Improved Algorithms for Network Topology Discovery,” *Proc. of Passive and Active Network Measurement (PAM)*, LNCS 3431, PP. 149-162, 2005.

[20] D. Xianhua, D. Zhenguo, and W. Qinqin, “Improved Algorithms for Large-Scale Topology Discovery,” *Proc. of IEEE International Symposium on Information Engineering and Electronic Commerce*, PP. 506-509, 2009.

[21] B. Li, J. He, and H. Shi, “Improving the Efficiency of Network Topology Discovery,” *Proc. of 3rd International Conference on Grid and Pervasive Computing Symposia*, PP. 189-194, 2008.

[22] Z. Pei-Dong, S. Huai-Zhou, and L. Xin, “An Enhanced Router Topology Discovery Model,” *Proc. of the 7th Annual Communications Networks and Services Research Conference*, PP. 431-433, 2009.

[23] H. Zhao, M. Chen, L. Song, and H. Bai, “A Novel Router Level Topology Discovery Algorithm,” *IEEE International Conference on Advanced Intelligent Mechatronics*, PP. 1412-1417, 2008.

[24] Y. Zhao, J. Yan, and H. Zou, “Study on Network Topology Discovery in IP Networks,” *Proc. of IEEE*, 2010.

[25] C. Son, J. Oh, K. Lee, K. Kim, and J. Yoo, “Efficient Physical Topology Discovery for Large OSPF Networks,” *IEEE/IFIP Network operations and management symposium: Pervasive Management for Ubiquitous Networks and Services*, PP. 325-330, 2008.

[26] G. Malkin, “Traceroute Using an IP Option,” IETF RFC-1393, January 1993.

[27] S. Deering, and R. Hinden, “Internet Protocol, Version 6 (IPv6)Specification,”IETFRFC-2460, December 1998.

[28] S. Hagen,“IPv6 Essentials,” O'Reilly Media, May 2006.

[29] A. Conta, S. Deering, and M. Gupta, “Internet Control Message Protocol (ICMPv6)for the Internet Protocol Version 6 (IPv6) Specification,” IETF RFC-4443, March 2006.

[30] J. Abley, P. Savola, and G. Neville-Neil, “Deprecation of Type 0 Routing Headers in IPv6,”IETF RFC-5095, December 2007.

[31] Common Open Research Emulator (CORE), [Online], available: http://www.nrl.navy.mil/itd/ncs/products/core/, last accessed on April 28th, 2014.

[32] Quagga Routing Suite, [Online], available: http://www.nongnu.org/quagga/, last accessed on April 28th, 2014.

[33] R. Graziani, and A. Johnson, “Routing Protocols and Concepts: CCNA Exploration Companion Guide,” Cisco Press, USA, 2008.

[34] R. Coltun, D. Ferguson, J. Moy, and A. Lindem, “OSPF for IPv6,” IETF RFC-5340, July 2008.

[35] A. Nogueira, and P. Ferreira, “A Practical Approach to Corporate Network Engineering,” River Publishers, Aalborg, Denmark, 2013.

[36] Wireshark, [Online], available: http://www.wireshark.org/, last accessed on April 17th, 2014.

[37] P. Oppenheimer, and J. Bardwell, “ Troubleshooting Campus Networks: Practical Analysis of Cisco and LAN Protocols,” Wiley Publishing, Inc., Indianapolis, Indiana, 2002, ISBN: 978-0-471-21013-9.

[38] J. Sloan, “Network Troubleshooting Tools,” O'Reilly Media, August 2001.

[39] J. Saltzer, D. Reed, and D. Clark, “Source Routing for Campus-Wide Internet Transport,” *Proc. of Local Networks for Computer Communications*, PP. 1-25, Zurich, Switzerland, August 1980.

[40] P. Harrison, “Linux Quick Fix Notebook,” Pearson Education Inc., USA, March 2005, ISBN-10: 0131861506.

[41] B. Green, and P. Smith, “Cisco ISP Essentials,” Cisco Press, USA, 2002, ISBN 1-58705-041-2.

[42] P. Killelea, “Web Performance Tuning,” O'Reilly and Associates, Inc., March 2002.

[43] T. Szigeti, C. Hattingh, R. Barton, and K. Briley, “End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks,” Cisco Press, USA, 2014, ISBN 978-0-13-311610-6.

[44] J. Tiso, “Designing Cisco Network Service Architectures (ARCH): Developing an Optimum Design for Layer 3,” Cisco Press, USA, 2011.

[45] J. Postel, “Internet Control Message Protocol,” IETF RFC-792, September 1981.

[46] A. Danesh, L. Trajkovic, S. Rubin, and M. Smith, “Mapping the Internet,” *Proc. of IEEE*, 2001.

[47] X. Zou, Z. Qiao, G. Zhou, and K. Xu, “A logic distance-based method for deploying probing sources in the topology discovery,” *Proc. of IEEE GLOBECOM*, 2009.

[48] W. Han, and K. Xu, “A method for placing traceroute-like topology discovery instrumentation ,” *Proc. of IEEE International Conference on Communication Systems*, PP. 1160-1164, 2008.

[49] J. Liu, Q. Wang, and J. Luo, “A New Distributed Topology Discovery Technology for IPv6 Networks ,” *Proc. of IEEE*, PP. 627-632, 2007.