University of Nevada, Reno

A Graph Theoretic Perspective on Internet Topology Mapping

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in
Computer Science and Engineering

by

Hakan Kardes

Dr. Mehmet H. Gunes / Dissertation Advisor

December, 2012

THE GRADUATE SCHOOL

We recommend that the dissertation
prepared under our supervision by

HAKAN KARDES

entitled

A Graph Theoretic Perspective On Internet Topology Mapping

be accepted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

Mehmet Hadi Gunes, Ph.D., Advisor

Eelke Folmer, Ph.D., Committee Member

Sergiu Dascalu, Ph.D., Committee Member

Murat Yuksel, Ph.D., Committee Member

Cahit A Evrensel, Ph.D., Graduate School Representative

Marsha H. Read, Ph. D., Dean, Graduate School

December, 2012

A Graph Theoretic Perspective on Internet Topology Mapping

Publication No. _____

Hakan Kardes

University of Nevada, Reno, 2012

Supervisor: Mehmet H. Gunes

Abstract

Understanding the topological characteristics of the Internet is an important research issue as the Internet grows with no central authority. Internet topology mapping studies help better understand the structure and dynamics of the Internet backbone. Knowing the underlying topology, researchers can better develop new protocols and services or fine-tune existing ones. Subnet-level Internet topology measurement studies involve three stages: topology collection, topology construction, and topology analysis. Each of these stages contains challenging tasks, especially when large-scale backbone topologies of millions of nodes are studied.

In this dissertation, I first discuss issues in subnet-level Internet topology mapping and review state-of-the-art approaches to handle them. I propose a novel graph data indexing approach to to efficiently process large scale topology data. I then

conduct an experimental study to understand how the responsiveness of routers has changed over the last decade and how it differs based on the probing mechanism. I then propose an efficient unresponsive resolution approach by incorporating our structural graph indexing technique.

Finally, I introduce *Cheleby*, an integrated Internet topology mapping system. Cheleby first dynamically probes observed subnetworks using a team of PlanetLab nodes around the world to obtain comprehensive backbone topologies. Then, it utilizes efficient algorithms to resolve subnets, IP aliases, and unresponsive routers in the collected data sets to construct comprehensive subnet-level topologies. Sample topologies are provided at `http://cheleby.cse.unr.edu`.

Acknowledgments

I would like to express my sincere gratitude for my adviser Dr. Mehmet H. Gunes, whose guidance, inspiration, understanding, patience, and encouragement added considerably to my graduate experience. I would like to thank Dr. Murat Yuksel, Dr. Eelke Folmer, Dr. Sergiu Dascalu, and Dr. Cahit Evrensel for agreeing to be on my dissertation committee despite their extremely busy schedule and for their invaluable comments.

I would like to thank my colleagues David Shelly and Talha Oz for their effort to put together the Cheleby project and all current and former Computer Networking Lab members, Abdullah Sevincer, Bilal Gonen, Engin Arslan, Esra Erdin, Hayreddin Ceker, Jeffrey Naruchitparames, Mehmet Bilgi, Mehmet Burak Akgun, Omer Kilavuz, Suat Mercan, and Tarik Karaoglu for all their support during my graduate studies. This dissertation also would never have been completed without the encouragement of my colleagues, Dr. Vitor Carvalho, Dr. Xin Wang, Dr. Andrew Borthwick, Yigit Kiran, and Jim Adler.

I would also like to thank my family for the support they provided me through my entire life. Finally, I must acknowledge Enes, one of my best friends, without whose encouragement I would not have decided an academic career.

Hakan Kardes

University of Nevada, Reno

December 2012

Contents

Abstract **i**

Acknowledgments **iii**

List of Tables **viii**

List of Figures **x**

Chapter 1 Introduction **1**

Chapter 2 Issues in Internet Topology Mapping and Literature Survey **7**

2.1 Topology Collection 9

2.1.1 Sampling Bias 9

2.1.2 Load Balancing 11

2.1.3 Probing Overhead 12

2.2 Topology Construction 13

2.2.1 Subnet Resolution 14

2.2.2 IP Alias Resolution 15

2.2.3 Unresponsive Router Resolution 18

Chapter 3 Structural Graph Indexing **23**

3.1 Introduction 24

3.2 Related Work 25

3.3 Structural Graph Indexing 26

3.3.1 Structure Models 26

3.4 Evaluation on Internet Topologies 34

3.4.1 Graph Transformation 34

3.4.2 Structure Statistics 36

3.4.3 Resolution Results 44

3.5 Evaluation on a Wikipedia Graph 44

3.6 Summary 46

Chapter 4 Unresponsive Routers **47**

4.1 Introduction 48

4.2 Router Responsiveness Analysis 52

4.2.1 Historical Data Analysis 53

4.2.2 Responsiveness to Different Probe Mechanisms 60

4.2.3 Summary 63

4.3 Graph Based Induction for

Unresponsive Router Resolution 64

4.3.1 Structural Graph Indexing 67

4.3.1.1 Parallel *-subpath Structures 67

4.3.1.2 Star Structures 70

4.3.1.3 Complete Bipartite Structures 72

4.3.1.4 Triangle Structure 73

4.3.2 Unresponsive Router Resolution 75
4.3.2.1 Triangle Resolution 75
4.3.2.2 Complete Bipartite Resolution 75
4.3.2.3 Star Resolution 77
4.4 Evaluations 77
4.4.1 Simulation-based Evaluations 77
4.4.2 Impact of Alias Resolution 80
4.4.3 Experimental Results 81
4.5 Summary 83

Chapter 5 Cheleby: An Internet Topology Mapping System 85

5.1 Introduction 86
5.2 Topology Sampling 86
5.2.1 Destination List Generation 87
5.2.2 Response Wait Time 88
5.2.3 Task Assignment to Monitors 88
5.2.4 Probing Overhead Reduction 98
5.3 Topology Construction 100
5.3.1 Initial Pruning 102
5.3.2 Subnet Inference 105
5.3.3 IP Alias Resolution 106
5.3.4 Unresponsive Router Resolution 108
5.3.5 Increasing Graph Density 113
5.4 Summary 116

Chapter 6 Internet Topology Mapping Systems **117**

6.1 Ark 117

6.2 Dimes 118

6.3 iPlane 120

6.4 Cheleby 122

Chapter 7 Conclusions **125**

Bibliography **128**

List of Tables

3.1 SGI for Resolving unresponsive Routers in iPlane Data-set 44

4.1 Analysis of Historical Responsiveness 54
4.2 *-Subpath Characteristics 56
4.3 Responsiveness to Direct Probes 58
4.4 Responsiveness to Indirect Probes 63
4.5 14% Unresponsive Router Regions for (10,2000) T-S Sample 78
4.6 Graph Based Induction Technique on Real Data Sets 79
4.7 Impact of Alias Resolution 80
4.8 Complexity and Operational Overhead of GBI 83
4.9 Size of the Data Structures 83

5.1 Team Statistics with Different Team Sizes 91
5.2 Team Statistics (Average of 8 Data Sets) 93
5.3 Topology Data (in millions) 104
5.4 Average Subnet Statistics for 8 Data Sets 106
5.5 Alias Resolution Averages 108
5.6 Unresponsive Router Resolution (Average of 8 data sets) 110
5.7 Improved Subnet Statistics 113

5.8 Improved Alias Resolution Statistics 115

6.1 Internet Topology Mapping Systems 124

List of Figures

1.1 Cheleby System Overview 6

2.1 Effect of Sampling Bias 9
2.2 Effect of Load Balancing 11
2.3 Effect of Subnet Resolution 14
2.4 Effect of IP Aliases 16
2.5 Effect of Unresponsive Routers 19

3.1 Structural Models 27
3.2 **Algorithm 1** - Star Structure Indexing 28
3.3 **Algorithm 2** - Complete Bipartite Structure Indexing 30
3.4 **Algorithm 3** - Triangle Structure Indexing 31
3.5 **Algorithm 4**: Clique Structure Indexing 33
3.6 Sample Transformation 34
3.7 Transformation of Graph in Figure 2.5-b 35
3.8 Star Structure Distributions 37
3.8 Star Structure Distributions 38
3.9 Bipartite Structure Distribution - 2006 39
3.10 Bipartite Structure Distribution - 2010 40

3.10 Bipartite Structure Distribution - 2010 41
3.11 Triangle Structure Distribution 42
3.12 Clique Structure Distribution 43
3.13 Wikipedia Graph 45

4.1 Sample Network 48
4.1 Sample Network 49
4.2 Active Probing 52
4.3 Historic Distance Distribution 59
4.4 Structures (genuine and observed) 65
4.5 **Algorithm 5**: Finding Parallel *-Subpaths 68
4.6 Resolution of Temporarily Unresponsive Routers 69
4.7 **Algorithm 6**: Resolving Temporarily Unresponsive Routers 70
4.8 **Algorithm 7** - Star Structure Indexing 71
4.9 **Algorithm 8** - Complete Bipartite Structure Indexing 72
4.10 **Algorithm 9** - Triangle Structure Indexing 74
4.11 **Algorithm 10**: Resolving Triangle Substructures 76
4.12 **Algorithm 11**: Resolving Complete Bipartite Substructures 76
4.13 **Algorithm 12**: Resolving Star Substructures 76
4.14 Degree distribution 82

5.1 Cheleby System Overview 86
5.2 Cumulative Distribution Function for RTT 88
5.3 Team assignment of PlanetLab nodes (Blue lines: 5 teams. Green boxes: 7 teams.) 90
5.4 Number of Nodes and Edges for different team sizes 92

5.5 Completion Time per Destination Block (in Seconds) 94
5.6 Team Completion Statistics 95
5.7 The Number of New Known Nodes After Each Vantage Point 97
5.8 The Number of New Unresponsive Nodes After Each Vantage Point . 97
5.9 The Number of New Edges After Each Vantage Point 98
5.10 Number of Nodes and Edges (average of 8 data sets) 99
5.11 Intra- and Inter-monitor Redundancy Reduction 100
5.12 Topology Construction 101
5.13 Sample Transformation 104
5.14 Subnet Resolution 105
5.15 Analytical and Probe-based Alias Resolver v2 (APARv2) 107
5.16 Number of Unresponsive Routers with Different Lengths 111
5.17 Initial vs. Final Unresponsive Routers 111
5.18 Initial vs. Final Unresponsive Routers according to UR length 112
5.19 Percentages of Resolved Unresponsive Routers after Each Step 112
5.20 Cheleby Subnet Distribution 114

6.1 Ark Router Interface Distribution 119
6.2 Ark Node Degree Distribution 120
6.3 iPlane characteristics 121
6.4 Cheleby characteristics 123

Chapter 1

Introduction

Internet has become a cornerstone of our daily life. The number of users has grown from a few researchers in early days to billions of population. As the largest man made complex network, Internet grows with no central authority. Thousands of small and medium size Autonomous Systems (ASes) connect individuals, businesses, universities, and agencies while focusing on optimizing their own communication efficiency. Even though communication protocols and networking devices have been analyzed in depth, the Internet as a whole has not been well characterized.

As each network is built by different AS for possibly different purposes, e.g., small local campus to large transcontinental backbone provider, a single AS is not representative of the whole Internet [102]. Each AS grows its network based on local economic and technical objectives. Moreover, even though Internet Service Providers (ISPs) may compete for customers, they need to cooperate to provide overall connectivity.

Internet topology maps are needed due to the commercial, social, and technical purposes. For instance, such graphs are valuable in analyzing the topologi-

cal characteristics of the Internet and designing topology generators that can produce realistic synthetic topologies. Additionally, analysis of the Internet topology is needed to develop failure detection measures, network planning, and optimal routing algorithms [64]. Researchers test new protocols and systems using simulations or emulations [8, 126], but more realistic results can be obtained when real topologies are utilized in the analysis [32, 71]. Network anomalies can also be identified using the topology measurements [76, 96, 131]. Furthermore, knowledge of the network graph helps in understanding the large scale characteristics and dynamics of the Internet [98]. Internet topology analysis also provides insight into the current trends. For instance, researchers have indicated that the deployment of networks by content providers has a flattening effect on the hierarchical autonomous system (AS) structure [50, 83]. Similarly, evolution of the topology can be analyzed to predict the future growth trends [63].

However, due to commercial and security reasons, ISPs keep their topology information confidential. This policy introduces challenges for the research community to generate measurement probes for sampling the Internet topology at various levels, including IP [89], router [114], point of presence (PoP) [45, 129], subnet [73, 121] and AS levels [66, 88]. Several research groups have developed mapping systems to collect the required information. These include the PlanetLab measurement infrastructure [8], the Archipelago measurement infrastructure of CAIDA [85], the iPlane infrastructure [87], the DIMES project [110], the Scriptroute project [116], and several others [92, 114, 122, 124].

Internet topology measurement studies, often, require three phases: (1) topology sampling, (2) topology construction, and (3) topology analysis. Inaccuracies in the first two processes may significantly affect the validity of the results obtained in

the measurement study [53, 58, 118]. Most measurement studies utilize *traceroute* [67] to collect a large number of path traces from topologically diverse set of vantage points. After collecting the path traces, the information is processed to build the corresponding topology map. In particular at subnet, router and PoP levels, following issues should be addressed:

- **Sampling Bias:** During topology collection, the measurement study should eliminate sampling bias [72, 84]. Since there are a limited number of vantage points and a large number of destinations, collected topology may be biased towards the vantage points.

- **Load Balancing:** Path tracing should consider the load balancing in order to obtain accurate traces. Certain traffic engineering practices for load balancing may cause traceroute to return IP addresses that do not correspond to a real end-to-end path in the network [21]. This happens when a router forwards consecutive traceroute probes on different paths toward the destination, a common phenomenon in the Internet [18].

- **Probing Overhead:** As the volume of active measurement practices has increased in time, it is important to minimize redundant probing and carefully consider any disruption that might be caused by the measurement study.

- **Unresponsive Router Resolution:** Certain routers are passive to measurement probes and are represented by a '*' in a traceroute output [61]. Since a router may appear as a '*' in multiple traceroute outputs, we need to identify '*'s (i.e., unresponsive nodes) that belong to the same router [60].

- **IP Alias Resolution:** As routers have multiple interfaces, each interface has a unique IP address. In a given set of path traces, a router may appear on multiple path traces with different IP addresses. Hence, we need to identify nodes that appear to be separate in collected path traces and combine them into a single node (i.e., to indicate IP addresses that belong to the same router) [62, 120].

- **Subnet Resolution:** As routers are connected to each other over point-to-point or multi-access links, subnet resolution helps in identifying the underlying link-level connectivity. Hence, we need to analyze observed IP addresses to identify subnets and combine observed links to represent the corresponding single hop connection medium (i.e., point-to-point or multi-access link) [59,119].

The accuracy and the completeness of these tasks may significantly affect the representativeness of the resulting topology maps [58]. Hence, topology measurement studies should address these tasks to obtain a representative topology map. Even thought there are several systems providing the raw Internet topology data, there is no system providing up-to-date Internet graph at subnet or router level.

In this dissertation, we first discuss the issues in subnet-level Internet topology mapping and review state-of-the-art approaches to handle them. Secondly, we present our novel *Structural Graph Indexing* (SGI) [74] approach for efficiently mining complex networks. As indexing feature, we utilize graph structures such as star, complete bipartite, triangle and clique that frequently appear in protein, chemical compound, and Internet graphs. SGI lists all substructures matching formulated structures while other structures can be identified and added to the SGI. Different from previous graph indexing approaches, SGI does not limit the number of nodes in the indexes and provides an alternative tool for querying large-scale graphs.

Next, we investigate the responsiveness of routers to active network measurements in two directions *(i)* historical router responsiveness to active measurements and *(ii)* current responsiveness to different probe mechanisms. Expanding an earlier work [61] on skitter [94] data sets, we present the prevalence of unresponsive routers in historical Ark [2] and iPlane [87] measurement data. For today's practices, we use different types of active probes, e.g, ICMP, UDP, and TCP, to observe the responsiveness of routers.

We then enhance the *Graph Based Induction* (GBI) approach of [60] to resolve unresponsive routers by incorporating our SGI technique and utilizing the findings from our measurement study. In this approach, we index observed subgraphs that contain unknown nodes and determine the corresponding minimal underlying structure that satisfies the trace accuracy condition. We also show that proper resolution of IP aliases improves the unresponsive router resolution. Our work improves the state of the art in unresponsive router resolution in terms of accuracy and efficiency.

We then introduce *Cheleby*[1] [73], an integrated Internet topology mapping system. Cheleby provides insight into the Internet backbone topology by taking daily snapshots of the underlying networks. For this, Cheleby assembles state-of-the-art topology collection and construction techniques, i.e., target list generation, probe redundancy reduction, unbiased accurate data collection, subnet inference, alias resolution, and unresponsive router resolution.

[1]Evliya Cheleby was an Ottoman Turkish writer who extensively travelled through the Ottoman empire and neighboring countries from 1640 to 1676. We named our topology mapping system after him as our system collects topology information by traversing different paths from geographically diverse points.

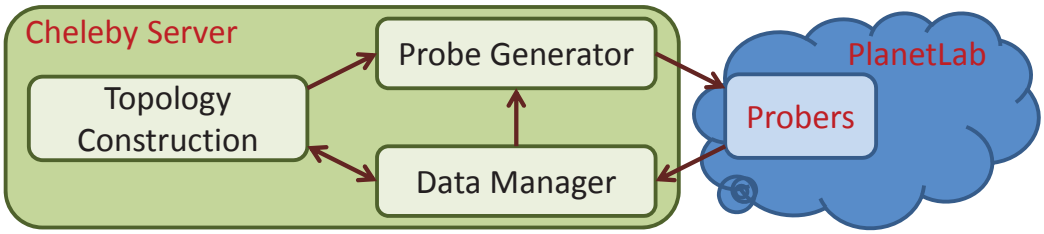


Figure 1.1: Cheleby System Overview

Cheleby topology mapping system, shown in Figure 1.1, runs on a server which actively manages PlanetLab nodes as its monitors to collect topology information from geographically diverse vantage points. The server instructs monitors to collect partial path traces and perform other probing activities. It then utilizes efficient algorithms for resolving subnets, IP aliases, and unresponsive routers in collected data sets to build accurate subnet-level topologies. Incorporating enhanced resolution algorithms for collected raw data, Cheleby provides comprehensive topology maps.

Overall, main contributions of this dissertation are:

- A novel *structural graph indexing* (SGI) approach for efficiently mining complex networks.
- A state-of-the-art unresponsive router resolution approach that is practical for large-scale data sets.
- Cheleby Internet mapping system, which assembles state-of-the-art topology collection and construction techniques.

Chapter 2

Issues in Internet Topology Mapping and Literature Survey

Understanding the topological characteristics of the Internet is an important research issue as the Internet grows with no central authority. Internet topology measurement studies help better understand the structure and dynamics of the Internet backbone. Additionally, Internet measurement studies can be divided into two categories: (i) *active* measurement studies that generate probes to elicit responses from systems in the network and (ii) *passive* measurement studies that do not generate traffic but just observe the system.

In recent years, several research groups have been increasingly using active probing to analyze various characteristics of the underlying Internet topology. For example, in [76, 94, 110], active probing has been used to analyze the routing and reachability behavior of the Internet. Additionally, several overlay or p2p network applications utilize active probing for the performance optimization of their applications [82, 97]. Moreover, Luckie et al. analyzed router responsiveness for TCP,

UDP, and ICMP based active probing in [86]. They found that reachability ratio of ICMP-based traceroute method is higher than that of the other ones. ICMP-based method also collects evidence of a greater number of AS links. On the other hand, UDP-based method infers the greatest number of IP links, despite reaching the fewest destinations.

Several large scale distributed Internet measurement platforms have been developed to facilitate topology measurement studies and conduct various measurement activities that include active probing. These include the PlanetLab measurement infrastructure [8], the Archipelago measurement infrastructure of CAIDA [85], the iPlane infrastructure [87], the DIMES project [110], the Scriptroute project [116], and several others [92, 114, 122, 124].

Internet topology measurement studies, often, require three phases: (1) topology sampling, (2) topology construction, and (3) topology analysis. Several studies have shown that inaccuracies in the first two processes may significantly affect the validity of the results obtained in the measurement study [53, 58, 118]. For topology collection, most measurement studies utilize *traceroute* [67] to collect a large number of path traces from topologically diverse set of vantage points. Traceroute is a well known network diagnostic tool which returns a path, list of routers, from a vantage point to a given destination by tracing the routers in between. After collecting the path traces, the information is processed to build the corresponding topology map.

In the remainder of this chapter[1], we review the related works in the Internet

[1]Preliminary versions of this chapter appeared in:

Hakan Kardes, Talha Oz, and Mehmet H. Gunes *Cheleby: A Subnet-level Internet Topology Mapping System*, COMSNETS'12, Banglore, India, Jan 3-7, 2012.

Hakan Kardes, and M.H. Gunes. *Subnet-level Internet Topology Mapping: Issues and Resolution Methodologies.*Elsevier Computer Networks. (under review)

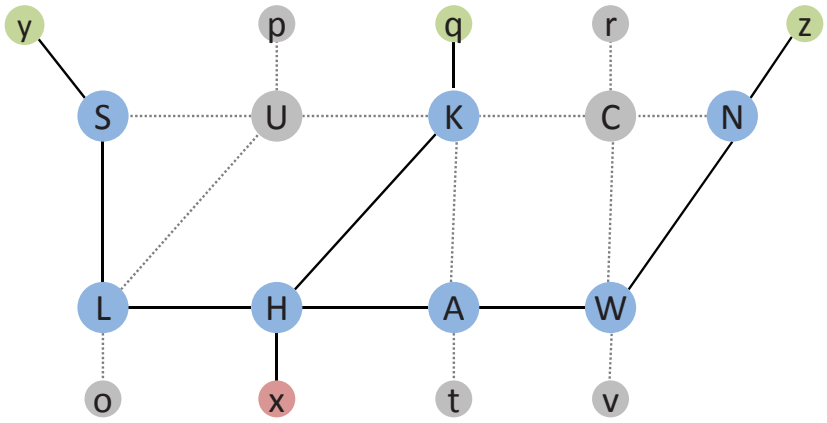


Figure 2.1: Effect of Sampling Bias

topology mapping area. Since, in this dissertation, we mainly focus on the topology collection and topology construction stages of the Internet topology mapping, we review these areas in more detail.

2.1 Topology Collection

There are several issues one needs to pay attention in sampling the Internet topology. In the following subsections, we discuss the effect of sampling bias, load balancers, and probing overhead.

2.1.1 Sampling Bias

An important issue in topology collection studies is to address sampling bias [72, 84] problem. In Internet topology sampling practices, usually, there are small number of vantage points from which traces originates and large number of destinations. Therefore, collected topology may be biased towards the vantage points. In such a case, connectivity of routers around the vantage point would be well discovered while many

distant routers and their connections would not be discovered as shown on Internet2 backbone in Figure 2.1. In the figure, orange node **x**, is the vantage point and green nodes **y**, **q** and **z** are trace destinations. In this case, only blue nodes, i.e., **S**, **K**, **N**, **L**, **H**, **A** and **W**, and corresponding links will be discovered revealing all neighbors of the vantage point but few of the destination nodes. Even though several researchers have pointed that adding new vantage points as traceroute sources has diminishing return [23], others have emphasized the necessity of increasing the number of the vantage points in certain cases. [42,52,110]. Moreover, it is important to have geo-diverse vantage points for accurate sampling of the topological characteristics [111].

In order to detect the sampling bias, Lakhina et al. have proposed a sampling bias test based on the degree distribution characteristics of the topology in question [84]. According to their study, if the measurement procedure used in topology collection is not biased, node degree of a given router in the topology should not change according to its hop distance to the vantage point(s).

In general, sampling approach may affect the observed characteristics of the network. For instance, use of (k,m)-traceroute probes where the number of vantage points k is much less than the number of destinations m on a non power-law topology produce sample topologies with power-law degree distributions [31]. Additionally, each network has a number of nodes that are interconnected according to some structural and functional relations. When sampling the topology, some of these relations might get broken or modified [53]. Therefore, one should carefully select the sampling methodology so that the resulting topology represents the original topology based on the characteristics in question.

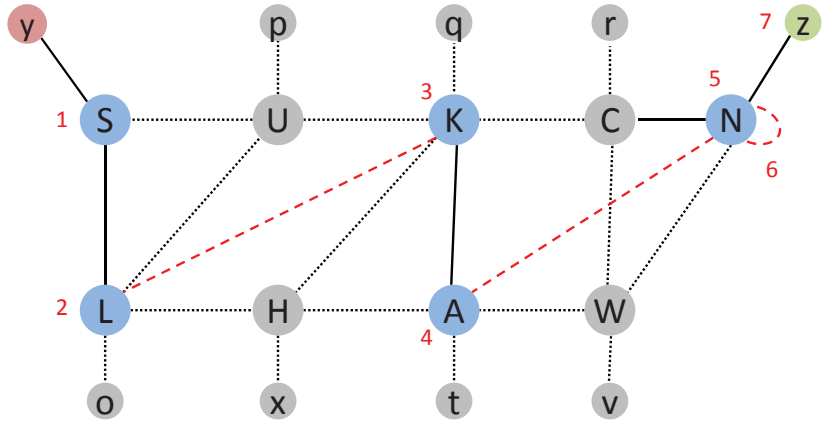


Figure 2.2: Effect of Load Balancing

2.1.2 Load Balancing

The second issue to keep in mind during topology collection is the deployment of load balancing by ISPs. Due to various traffic engineering practices for load balancing, traditional traceroute might return IP addresses that do not correspond to a real end-to-end path in the underlying network topology [21]. In general, load balancing happens when a router forwards sequential traceroute probes on different paths toward the destination [18]. For instance, consider a Internet2 backbone in Figure 2.2 where there are two paths between **y** and **z** (underlying links are marked with black lines). When router **S** forwards packets through both paths towards destination **z**, then path traces will include non-existent links (marked with dashed red lines) as we would observe a path of **S-L-K-A-N-N-z** from **s**.

Routers allow configuration of several parameters to determine load balancing including destination IP, source IP, protocol and port number [11, 12]. Additionally, routers can be set to randomly send packets as flow preserving parameters consume storage and computation resources. Augustin et al. classify load balancers as *per destination*, i.e., only destination IP affects load balancing decision, *per flow*, i.e., any

combination of the parameters are utilized for load balancing decision, and *per packet*, i.e., packets are randomly distributed [20].

Traditional traceroute tools couldn't address all types of load balancing. Even though they would find accurate links with per destination load balancer, they could not address per flow and per packet load balancers. On the other hand, *Paris traceroute* fixes flow identifiers by ensuring that per flow load balancing routers always pick the same next hop for probe packets toward the same destination [20]. However, *Paris traceroute* can only detect the existence of per packet load balancers on a path and would fail to accurately identify actual links on the balanced paths. Similarly, *sidecar* enables the record route option of probe packets to detect changes in traversed paths [113]. However, *sidecar* does not ensure that routers use the same path in forwarding probe packets. Hence, accurate path traces would not be collected.

2.1.3 Probing Overhead

Since active probing has been increasingly utilized in Internet measurement studies, it is important to minimize redundant probing so that Internet measurement studies do not disrupt the network. To this end, several researchers have presented approaches to reduce the amount of redundant active probes in measurement studies. For example, *Doubletree* prunes redundant probes to nodes that are close to both the vantage point and the destination since traces from a vantage point yield tree-like structures (i.e., trees rooted at the vantage point and at the destination, respectively) [41, 42]. Moreover, Jin et al. utilize a network coordinate system to identify the minimum set of path traces to collect at each vantage point [69]. Similarly, in order to reduce the number of probes, *AROMA* analyzes the convergence of path traces in discovering network topology [81]. More recently, Eriksson et al. proposed Depth-First Search

Ordering to reduce the probing overhead by clustering destinations based on a shared infrastructure [44]. Finally, routing underlay has been proposed to unify measurement activities [97]. In this method, overlay networks query the routing underlay, which aggregates collected topology information.

2.2 Topology Construction

After collecting topology data one needs to process this information to obtain the underlying network topology [39, 117]. Obtaining an accurate network map requires several important tasks including: (1) filter faulty traces, e.g., initial pruning, (2) identifying underlying physical subnets among the IP addresses to obtain the subnet level connectivity [59], (3) finding IP addresses belonging to the same router as routers may appear with different IP addresses in different path traces [26, 54, 62], and (4) resolving unresponsive routers that are represented by '*'s in probing output since some routers do not respond to probes during topology collection [60, 128]. These resolution tasks, shown in Figure 5.12, are especially challenging when large scale topologies of millions of nodes are processed.

The accuracy and the completeness of topology construction tasks significantly affect the accuracy of the resulting topology maps [58,118]. Moreover, when handling these tasks, one needs to make decisions based on observations from the measurement data. As the earlier decisions affect the later ones, obtaining the most likely topology under various conditions has shown to be NP-hard [14]. Hence, several approaches have been proposed to reduce the set of hypotheses in the decision making of the resolution tasks [19, 60, 62, 100, 112].

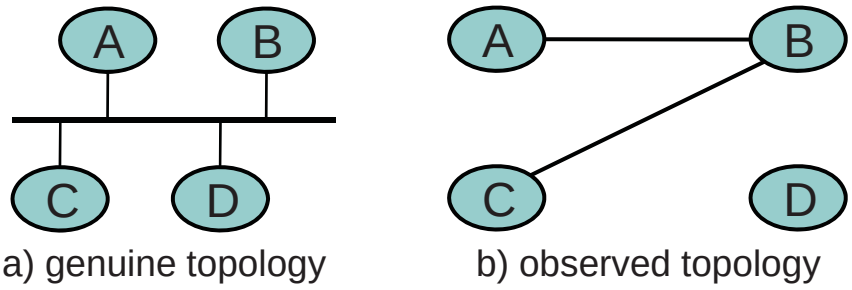


Figure 2.3: Effect of Subnet Resolution

In this section, we analyze each of these topology construction issues, and discuss the approaches proposed by the research community. Note that, while these algorithms were verified on sample networks, complete verification is not possible since it requires the availability of the underlying topology map of the Internet.

2.2.1 Subnet Resolution

First task after building an initial network graph is the identification of the underlying physical subnets, i.e., the subnet level connectivity, among IP addresses in the collected topology [59]. Routers are connected to each other over subnetworks and the subnet resolution helps in identifying the underlying multi-access links. In this task, the IP addresses in a data set are analyzed to infer the subnet relations among them. As an example, consider four routers A, B, C, and D, in Figure 2.3-a, that are connected to each other via a multi-access link. Assume that a collected set of path traces includes A-B link and B-C link and no path trace at hand includes the A-C link or any link between D and other routers from the subnet. In this case, a subnet level map that does not consider the subnet relation among these IP addresses will yield in a subgraph as shown in Figure 2.3-b which is considerably different from the

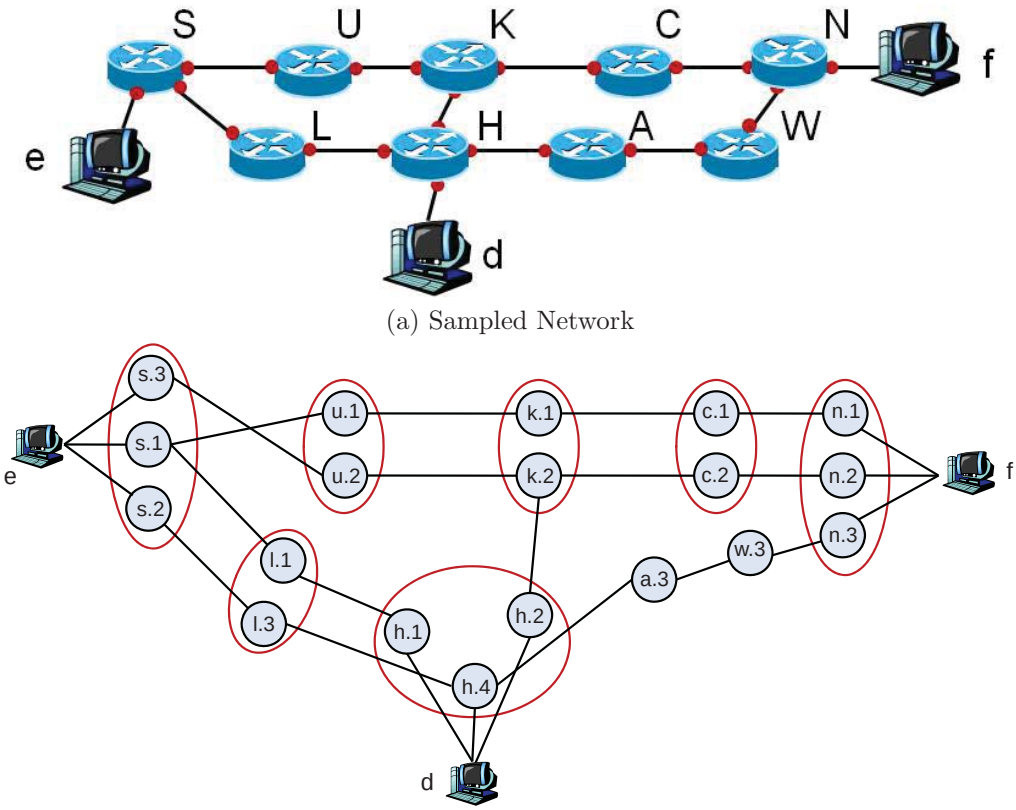
underlying topology. On the other hand, a careful study of the IP addresses may detect the subnet relation between the routers and therefore improve the resulting map with inclusion of the missing links.

The goal in subnet resolution is to identify multiple links that appear to be separate and combine them to reveal their corresponding single hop connection medium. This is similar to the IP alias resolution task where the goal is to identify nodes that appear to be separate in collected path traces and combine them into a single node (see Section 2.2.2). Subnet resolution also finds missing links between IP addresses that belong to the same subnet range but were not observed in collected path traces. The successfully inferred subnet information helps in improving the scope and quality of the resulting map by annotating it with additional information, i.e., the subnet relations among the existing set of IP addresses. Hence, inclusion of subnet relations among the routers yields topology maps that are closer, at the subnet layer, to the sampled segments of the Internet.

In this direction, in [59], Gunes et al. identified the subnet resolution task for topology construction studies and analyze its utility. Moreover, TraceNET tool enhances traceroute by identifying subnets between a source and a given destination in order to collect more complete and accurate topology maps [119]. TraceNET, however, focuses on a single path and infers subnets on an end-to-end path. Furthermore, ExploreNET discovers individually targeted subnets [121] . Authors also analyze the statistical biased versus unbiased sampling of subnets.

2.2.2 IP Alias Resolution

The second issue to consider during subnet-level topology construction is the IP aliases. As routers have multiple interfaces, each interface has a unique IP address.



(a) Sampled Network

(b) Resulting Network

Figure 2.4: Effect of IP Aliases

In a given set of path traces, a router may appear on multiple path traces with different IP addresses. Therefore, there is a need to identify and group IP addresses belonging to the same router. Without IP alias resolution the resulting topology map might be significantly different from the underlying topology [57,58]. For instance, in Figure 2.4-a, each router has multiple interfaces with unique IP addresses. Collecting traces between all pairs of **e**, **d**, and **f** end systems, we would obtain a sampled topology as in Figure 2.4-b without the alias IP address resolution. We need to identify IP aliases for each router and cluster them as shown with red circles.

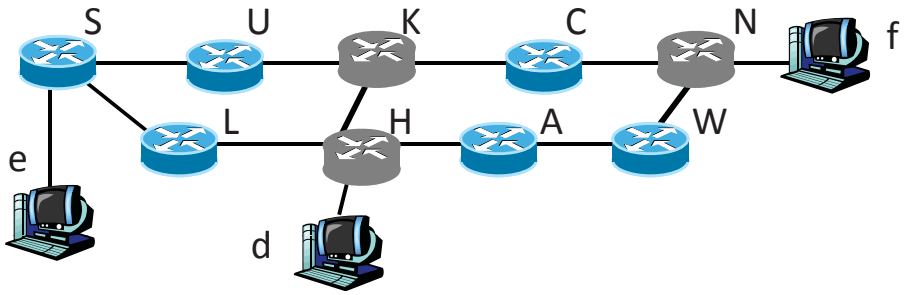
Several studies have emphasized the impact of incomplete IP alias resolution in certain measurement studies [27,118]. In [58], Gunes et al. performed an experimental study on the impact of IP alias resolution on various topological characteristics. Varying the resolution success rate, they analyzed various graph characteristics including topology size, node degree, degree distribution, joint degree distribution, characteristic path length, betweenness, and clustering. The results indicate that the IP alias resolution process has a significant impact on almost all topological characteristics that they consider. According to these studies, it can be concluded that Internet measurement studies should properly resolve the IP aliases to increase the accuracy of the final topology.

Moreover, several mechanisms have been proposed to resolve IP aliases [30, 51, 101, 113, 115] and few tools have been developed including ally [1] and iffinder [5]. These tools use an active probing approach to resolve IP aliases. They are easy-to-use and provide a convenient way to verify if a given pair of IP addresses is alias or not. As Gunes et al. stated in [57, 58], these tools requires participation of the routers by responding to the queries directed to them. This requirement introduces limitations to the success of IP alias resolution task since some routers are configured to ignore active probes directed to them. As an example, in their experiments, they observed that 40 percent of IP addresses that they probed with ally did not return a response. Hence, they presented a new IP alias resolution algorithm called Analytic and Probe-based Alias Resolver (APAR) [56]. APAR consists of an analytical component [57,58] and a probe-based component. Given a set of path traces, the analytical component utilizes the common IP address assignment scheme to infer IP aliases. Later, CAIDA released *kapar* which is an optimized implementation of APAR [79]. Furthermore, since ally requires $O(n^2)$ probes to test all possible pairs, several approaches have

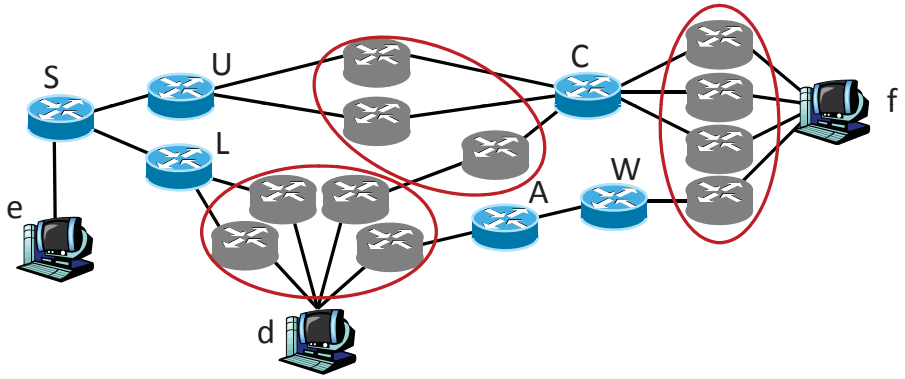
been deployed to reduce the number of probes [26, 79]. For example, MIDAR [79], Monotonic ID-Based Alias Resolution tool inspired by ally, provides greater precision, and sensitivity in a scalable manner. Moreover, as some routers do not respond to certain probes, Garcia et al. has proposed alternative probes to increase the elicited responses from routers [48].

2.2.3 Unresponsive Router Resolution

Unresponsive routers are the routers that are unresponsive to traceroute probes and are represented by a '*' in a traceroute output. Since an unresponsive router may appear in multiple traceroute outputs as '*', it is needed to identify '*'s (i.e., unresponsive nodes) that belong to the same router. Based on the number of unresponsive routers in the topology and the way of topology collection, there would be huge number of '*' in the collected set of path traces. For example, daily Internet topologies collected by Cheleby have more than 7M unresponsive nodes along with 1.2M known interfaces. Moreover, a sample network is shown in Figure 2.5-(a). Assume that in this topology H, K, and N are configured to act as unresponsive routers. When we run traceroute queries between all vantage points, represented as e, d and f in this figure, collected path traces will be as shown below. Using these traces, the resulting topology presented in Figure 2.5-(b).



(a) Sampled Network



(b) Resulting Network

Figure 2.5: Effect of Unresponsive Routers

```
d - * - L - S - e

d - * - A - W - * - f

e - S - L - * - d

e - S - U - * - C - f

f - * - C - * - * - d

f - * - C - * - U - S - e
```

Therefore, it can be concluded that even small number of unresponsive routers may significantly alter the final network topology and prevent researchers from obtaining the actual Internet topology from the trace paths collected. This example also shows the significance of the unresponsive router resolution task to obtain the actual topology map.

Unresponsive router resolution is an essential problem in topology mapping studies. Early work in this area did not pay attention to this problem or just proposed simple heuristics to address it [27, 28, 33]. Cheswick et al. stop the trace whenever they observe an unresponsive router on the path to the destination [33]. Broido et al. proposed two different approaches to address the issue [28]. First, they replace the unresponsive routers with edges to connect the known nodes at both ends of the unresponsive routers. In their second approach, they treat each occurrence of '*' as separate node. Bilir et al. addressed the issue by merging same length chain of unknown nodes between the same pair of known nodes with each other [27] (called as initial pruning in Sec. 4.4.1). However, all these approaches have some weaknesses or just provide very limited resolution like in [27](see Sec. 4.4.1). For example, [33] might result in losing significant connectivity information. Moreover, [28] causes inaccuracies in the final constructed maps.

Yao et al. analyzed the unresponsive resolution problem in detail and provided a formal definition to it [128]. In their study, they try to construct the minimum size topology by merging unknown nodes when they satisfy trace and distance preservation conditions. In *trace preservation condition*, merging two unknown nodes should not create a routing loop. For *distance preservation condition*, the length of shortest path between any given two nodes should not change at the end of resolution process. Next, they proved that merging unresponsive routers while satisfying above conditions is NP-complete problem. So, they proposed a heuristic to address the issue. However, their approach has a high algorithmic complexity of $O(n^5)$ which makes it impractical for large data sets.

Moreover, an ISOMAP based dimensionality reduction approach uses link delays or node connectivity as attributes to cluster nearby nodes [70]. The main lim-

itation of this approach is again its high complexity, i.e., $O(n^3)$ where n is the size of the topology. Additionally, they ignore the challenge in estimating individual link delays from round trip delays in path traces [46]. These two limitations makes the link delay based approach impractical for large scale topologies. They also provide a simple and more practical neighbor matching heuristic with $O(n^2)$ complexity but this approach might result in inaccuracies in the final graph with high false positive and false negative ratios(see Sec. 4.4.1).

Gunes et al. [60] added a new dimension to the problem by classifying the unresponsiveness types. They observe five different scenarios that cause routers to stay unresponsive. They then formulate a number of graph structures that can be found in traceroute-based topologies collected from the Internet. Namely, Parallel/Symmetric *-substrings, Clique, Complete Bipartite, and Star structures are defined. Based on this formulation, they introduced a graph based induction technique where they search for structures similar to the identified ones in the topology and then reduce the occurrences of unresponsive nodes into their corresponding routers. Even though this approach results in more accurate maps, its time complexity might be considerably reduced by using a proper graph indexing approach.

Finally, Almog et al. proposed semi-supervised spectral embedding of all nodes followed by clustering of the unknown nodes in the projected space [15]. As the authors indicate, the approach should be improved to be practical in large-scale data sets. Moreover, it assumes that there is only one unresponsive router between two known routers. According to our analysis in Table 4.2, however, there are considerable amount of unresponsive router chains and the number of these chains has been increasing in recent years. Additionally, they use a k-means clustering approach to cluster the unresponsive routers which requires the knowledge of the number of unre-

sponsive routers in the graph. Authors approximate the number of the unresponsive routers in the graph but, according to our study in Section 4.2, the number of unresponsive routers and their distribution depends on the topology collection approach. Thus, developing a general approximation technique which will work for different topology maps is a challenge.

Chapter 3

Structural Graph Indexing

Systems such as proteins, chemical compounds, and the Internet are being modeled as complex networks to identify local and global characteristics of the system. In many instances, these graphs are very large in size presenting challenges in their analysis. Hence, graph indexing techniques are developed to enhance various graph mining algorithms. In this chapter[1], we propose a new Structural Graph Indexing (SGI) technique that does not limit the number of nodes in indexing to provide an alternative tool for graph mining algorithms. As indexing feature, we use common graph structures, namely, star, complete bipartite, triangle and clique, that frequently appear in complex networks such as protein, chemical compound, social network, and Internet graphs. Note that, SGI lists all substructures matching structure formulations and other graph structures can be identified and added as indexing features.

[1] A Preliminary version of this chapter appeared in:

Hakan Kardes, M.H. Gunes. *Structural Graph Indexing for Mining Complex Networks.* IEEE ICDCS 2010 SIMPLEX, Genoa, ITALY, June 21 2010.

3.1 Introduction

Many systems can be modeled as a complex network to understand local and global characteristics of the system. Studying network models of systems provides a new direction towards better understanding biological, chemical, technological or social systems. In many cases, the systems under investigation are very large and the corresponding graphs have large number of nodes/edges requiring graph mining techniques to derive information from the graph. Several graph mining techniques have been developed to extract useful information from graph representation and analyze various features of complex networks [37]. In order to speed up graph queries, usually an index of the graph is derived according to some predefined index features.

Graph indexing is often utilized by graph search algorithms that look for a sub-graph within a graph database. For example, given a graph database G=$\{g_1, g_2, ..., g_n\}$ and a subgraph s, we are interested in identifying all graphs g_i that contain the subgraph s. This query is shown to be NP-complete [49] and becomes challenging as the size of graphs increase. For example, a typical graph of router-level Internet consists of millions of nodes making it impractical to perform many operations on the whole graph. In such cases, graph indexing allows operations to be more efficient.

In this chapter, we propose a new structural indexing approach to provide an alternative tool for graph mining algorithms. For indexing, we specify a set of common graph structures such as star, complete bipartite, triangle and clique. These structures are ubiquitous in biological, chemical, technological, and social networks. In order to reduce computational complexities, we index these structures within the original graph in a consecutive manner. We first identify star structures, and then the complete-bipartite, triangle and clique structures from the preceding ones. Our

approach, unlike the earlier indexing mechanisms, does not limit the size of subgraph considered in indexing. However, it may be limited as maximum clique search is an NP-complete problem [49].

In the remainder of the chapter, we first present existing graph indexing methods in Section 3.2. Next, we detail our structural graph indexing approach in Section 3.3. Then, we present an evaluation of the approach on Internet topologies in Section 3.4, and on a Wikipedia graph in Section 3.5. Finally, we conclude in Section 3.6.

3.2 Related Work

The need for mining large graphs in an efficient manner increases as researchers look into complex networks. Several studies have been carried out to make graph mining in an efficient manner using indexing techniques [34, 38, 68, 77, 95, 109, 127]. Many of the tools have constraints that limit the number of nodes/edges in index graphs or are not capable of operating on very large graphs. The graph indexing studies can be mainly categorized into two categories, namely, *path-based* and *structure-based* approaches.

Path-based graph indexing approaches use path expressions as indexing features such as GraphGrep [109] and Daylight [68]. GraphGrep enumerates all paths in the graph up to the length $maxL$. Then, it looks for each g_i whether it contains all paths up to $MaxL$ for a graph query q_i. A significant feature of path-based approaches is that paths can be manipulated more easily than general graphs. However, as Yan et. al. indicated, path is a simple structure loosing structural information of a graph, and hence false positive ratio of path-based methods would be very high [127].

In addition, the number of paths in a graph database increases exponentially making path-based methods impractical for very large graphs.

Alternatively, structure-based graph indexing approaches identifies subgraphs to be indexed as in gIndex [127]. gIndex first searches for the frequent subgraphs in the graph, then indexes these frequent structures. An issue in this case is that frequent subgraph discovery increases complexity and exponential number of frequent fragments may exist under low frequency support. Therefore, in their study, they limit the number of nodes and index frequent structures up to 10 nodes.

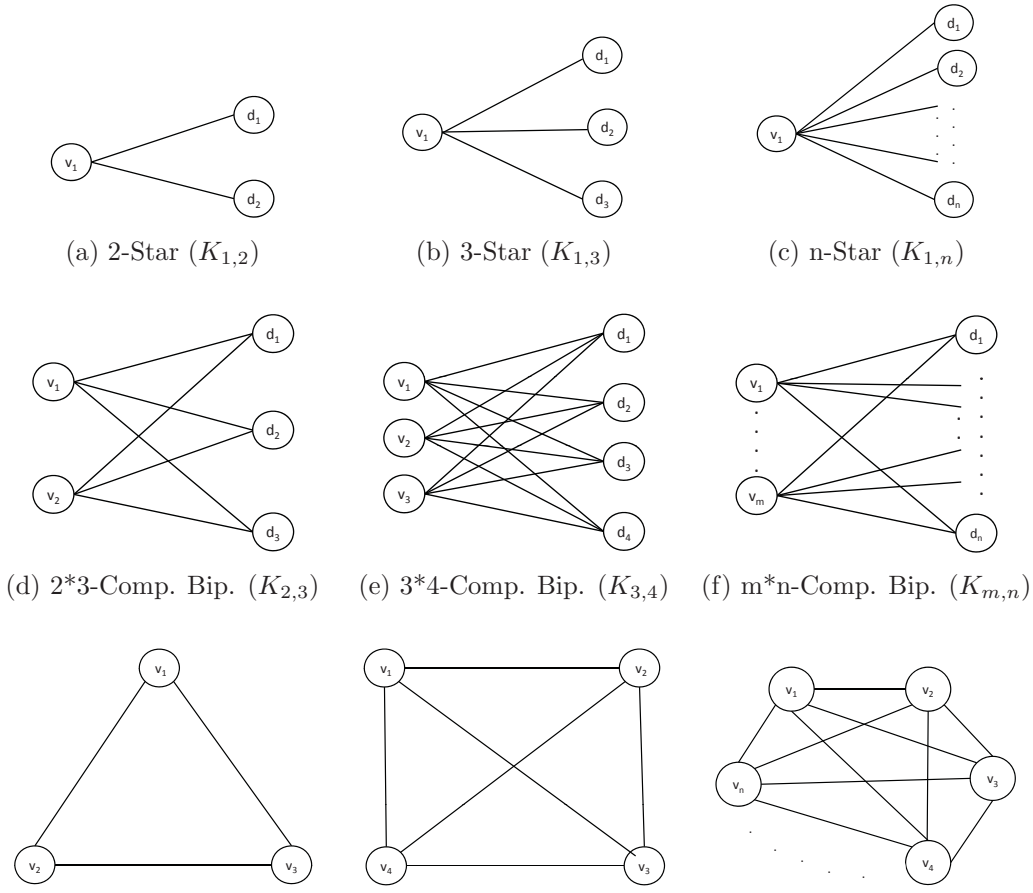
In this chapter, we propose an alternative structural indexing approach to search and process queries efficiently even in very large complex networks. As indexing features, we use commonly observed graph structures: star, complete bipartite, triangle and clique. An important feature of these structures is that each one is comprised from the previous one where clique contains complete bipartite structures and complete bipartite contains star structures.

3.3 Structural Graph Indexing

In this section, we present our structural graph indexing approach.

3.3.1 Structure Models

In structural indexing, we index predefined structures that are commonly observed in complex networks. In particular, we index star, complete bipartite, triangle and clique structures (shown in Figure 3.1) in a given graph $G = (V, E)$. An important difference of our approach from the previous studies is that we do not limit the size of subgraph considered in indexing. We index all maximal graphs that match the



(a) 2-Star ($K_{1,2}$) (b) 3-Star ($K_{1,3}$) (c) n-Star ($K_{1,n}$)

(d) 2*3-Comp. Bip. ($K_{2,3}$) (e) 3*4-Comp. Bip. ($K_{3,4}$) (f) m*n-Comp. Bip. ($K_{m,n}$)

(g) 3-Clique (Triangle) (K_3) (h) 4-Clique (K_4) (i) n-Clique (K_n)

Figure 3.1: Structural Models

structure formulation. For instance, a maximal clique is a clique that cannot be extended by adding one more vertex from the graph. However, the substructure size in indexing may be limited when needed since maximal clique search is known to be NP-complete [49]. In order to reduce computational complexities, we index the structures within the original graph in a consecutive manner. That is, we first identify star structures, and then the complete-bipartite, triangle and clique structures from the preceding ones as detailed below.

Let $G = (V, E)$; $S \leftarrow \phi$;
for (each *node* $v \in V$)
  $S \leftarrow S \cup s_{(v,\phi)}$
for (each *edge* $e(a, b) \in E$)
  $s_{(a,ns)} \leftarrow s_{(a,ns \cup \{b\})}$
  $s_{(b,ns)} \leftarrow s_{(b,ns \cup \{a\})}$
for (each $s_{(v,ns)} \in S$)
  if $|ns| < 2$
    $S \leftarrow S - s_{(v,ns)}$

Figure 3.2: **Algorithm 1** - Star Structure Indexing

Structure 1: Star ($K_{1,n}$)

We first index the star structure where a node has multiple neighbors as shown in Figure 3.1-(a), (b), and (c). All star structures within a graph $G = (V, E)$ are represented as $s(v_i, ns_i)$ where $v_i \in V$ and ns_i is the set of all neighbors of v_i. In graph theory, the star structure is utilized in the definition of some other structures and problems. A star with 3 edges is called a *claw*, as shown in Figure 3.1-(c). Claws are notable in the definition of claw-free graphs, graphs that do not have any claw as an induced subgraph. Additionally, a star is a special kind of tree, and several graph invariants are defined in terms of stars. Star arboricity is the minimum number of forests that a graph can be partitioned into such that each tree in each forest is a star [65], and the star chromatic number of a graph is the minimum number of colors needed to color its vertices in such a way that every two color classes together form a subgraph in which all connected components are stars [47]. Moreover, in [106] Robertson et al. showed that the graphs of branch-width 1 are exactly the graphs in which each connected component is a star.

We index maximal star structures for each node using the algorithm in Figure 3.2. The algorithm first builds a star structure $s_{(v_i,\phi)}$ for each node $v_i \in V$ without

any neighbors. Then, for each edge $e(a, b)$, it appends node a to the neighbor set of node, b and vice versa. Finally, the algorithm removes the star structures $s_{(v_i,ns_i)}$ such that the neighbor set ns_i has less than two neighbors. The overall run time complexity of this algorithms is $O(|V| + |E|)$.

Structure 2: Complete Bipartite ($K_{m,n}$)

The second structure we index is the complete bipartite, shown in Figure 3.1-(d), (e) and (f). A bipartite graph is a graph whose vertices can be divided into two disjoint sets V_1 and V_2 such that every edge connects a vertex in V_1 to one in V_2; that is, V_1 and V_2 are independent sets. A complete bipartite graph $G = (V_1 \cup V_2, E)$ is a bipartite graph such that V_1 and V_2 are two distinct sets and for any two vertices $v_i \in V_1$ and $v_j \in V_2$, then there is an edge between them (i.e., $\exists\ e^*_{(v_i,v_j)} \in E$). The complete bipartite graph with partitions of size $|V1| = m$ and $|V2| = n$ is denoted as $K_{m,n}$. Note that, star structure is a special case of a complete bipartite graph where $m = 1$. Moreover, finding complete bipartite subgraph $K_{m,n}$ with maximal number of edges $m.n$ is an NP-complete problem [104].

Complete bipartite structure is ubiquitous in many complex networks. For example, Thomas et al. examine the structure of protein-protein interaction networks and showed that the graph of all protein-protein interactions is made up of complete bipartite structures containing two disjoint sets of nodes in which each node in one set is connected to every node in the other set. [13]

We index all complete bipartite structures in the graph G using indexed star structures as in Figure 3.3. In the algorithm, for each star structure $s_{(a,ns)}$ where ns is the neighbors of the node a, we identify the maximal complete bipartite involving the node a. For this purpose, we find second hop neighbors of a by iterating over the

INPUT: S from Algorithm 1 in Figure 3.2
Let $G = (V, E)$; $K \leftarrow \phi$
for (each $s_{(a,ns)} \in S$)
  $L_{can} \leftarrow \phi$
  for (each $b_i \in ns$) where $b_i > a$
    $L_{can} \leftarrow L_{can} \cup ns^*$ where ($\exists\ s_{(b_i,ns^*)} \in S$)
  $L_{can} \leftarrow L_{can} - \{a\}$
  $R_{can} \leftarrow ns$

  for (each $v_i \in L_{can}$)
    $R_{new} \leftarrow R_{can} \cap ns_i^+$ where ($\exists\ s_{(v_i,ns_i^+)} \in S$)
    if ($|R_{new}| \geq 2$)
      $L_{new} \leftarrow \{a\} \cup \{v_i\}$
      for (each $v_j \in L_{can}$)
        if ($R_{new} \subset ns_j^{\#}$) where ($\exists\ s_{(v_j,ns_j^{\#})} \in S$)
          $L_{new} \leftarrow L_{new} \cup \{v_j\}$
        else if ($|R_{new} \cap ns_j^{\#}| \geq 2$) where ($\exists\ s_{(v_j,ns_j^{\#})} \in S$)
          $K \leftarrow K \cup k_{(L_{new} \cup \{v_j\}),(R_{new} \cap ns_j^{\#})}$
      $K \leftarrow K \cup k_{(L_{new},R_{new})}$

Figure 3.3: **Algorithm 2** - Complete Bipartite Structure Indexing

ns set and unifying them under L_{can} set that indicates candidates for the left side of the complete bipartite while the ns set is the candidate set for the right hand side. Then, we first find a $K_{2,n}$ and then grow it to $K_{m,n}$. In finding $K_{2,n}$, we iterate over each candidate node in the L_{can} as pivot node and determine the neighbor intersection with a. If the intersection set is larger than two, then these nodes belong to the right hand side. In the second step, we grow the $K_{2,n}$ by finding all nodes in the left hand side (i.e., L_{can}) that has the right hand side nodes (i.e., R_{new}) as a neighbor. Finding $K_{2,3}$ and larger complete bipartite graphs takes $O(|S|.|n_s|^2)$ where $|S|$ is the number of star structures in the graph, and n_s is the average node degree in the graph.

INPUT: S from Algorithm 1 in Figure
Let $G = (V, E)$; $T \leftarrow \phi$; $TS \leftarrow \phi$
for (each $s_{(a,ns)} \in S$)
  for (each $ns_i \in ns$) where $ns_i > a$
    if ($\exists$ $s_{(ns_i,ns_2)}$)
      $TS \leftarrow ns \cap ns_2$
      for (each $ts_i \in TS$) where $ts_i > ns_i$
        $T \leftarrow T \cup t_{(a,ns_i,ts_i)}$

Figure 3.4: **Algorithm 3** - Triangle Structure Indexing

Structure 3: Triangle (K_3)

Third, we index the triangle structure which is a clique structure of three nodes as shown in Figure 3.1-(f). Finding the number of triangles in a graph has become an important task over the last years due to its significant role in analyzing the density of complex networks. Tsourakakis et al. points to the importance of triangle counting [123]. Several commonly used complex network metrics such as the clustering coefficient and the transitivity ratio require the execution of a triangle counting algorithm.

Additionally, several interesting graph mining applications depend on computing the number of triangles in the graph [93], [99], [125], [22], [24], [43]. Especially in social networks, triangle is a well studied subgraph (i.e. motif). There are two main theories, namely the homophily and the transitivity, according to which triangles are generated in social networks. According to the homophily, people tend to choose friends that are similar to themselves. According to the transitivity, people who have common friends tend to become friends themselves [125]. Additionally, several other studies have showed that triangles play significant roles in many graph mining applications. For instance, Becchetti et al. showed that triangles can be used to detect spamming activity [24]. Similarly, Eckman et al. showed how triangles

can be used to uncover the hidden thematic structure of the web [43]. Additionally, triangle counting can benefit the query plan optimization in databases [22]. Thus, fast triangle counting algorithms are of high practical value.

We index all triangles in the graph by iterating over the star structures as in Figure 3.4. In the algorithm, for each star structure $s_{(a,ns)}$, and $s_{(ns_i,ns^*)}$ where $ns_i \in ns$, we take the intersection set TS of the sets ns and ns^*. Thus, for each $ts_i \in TS$, (a, ns_i, ts_i) constitutes a triangle. Indexing all triangles takes $O(|S|.log|S|.|n_s|)$ where $|S|$ is the number of star structures, and $|n_s|$ is the average node degree in the graph. Note that, processing edges connected to smaller degree nodes first may reduce the run time even though the time complexity remains the same.

Structure 4: Clique (K_n)

Finally, we index clique structures shown in Figure 3.1-(g) and (h). A clique in graph $G = (V, E)$ is a subset of the vertex set (i.e., $C \subseteq V$) such that there are edges between all node pairs (i.e., $\forall(c_i, c_j) \in C, \exists e_{(c_i,c_j)} \in E$, when $i \neq j$).

This structure has been observed in many fields. For example, in computational biology many problems can be solved by finding maximal or all cliques within the graph [25]. Similarly, Samudrala et al. models protein structure prediction as a problem of finding cliques in a graph whose vertices represent positions of subunits of the protein [108]; Cong et al. finds a hierarchical partition of an electronic circuit into smaller subunits using cliques [36]; and Rhodes et al. uses cliques to describe chemicals in a chemical database that have a high degree of similarity with a target structure [105].

We index all maximal clique structures (that has more than three nodes) in the graph using complete bipartite structures as in Figure 3.5. We first get the set

INPUT: K from Algorithm 2 in Figure 3.3
Let $G = (V, E)$; $C \leftarrow \phi$
for (each $k_{(m,n)} \in K$)
  for (each $a \in k_{(m,n)}$)
  findCliques($\{a\}, k_{(m,n)} - \{a\}$)

FUNCTION findCliques(L1, L2)
  if ($|L2| = 0$) and ($|L1| > 3$)
    $C \leftarrow C \cup c_{(L1)}$
  else
    for (each $b \in L2$)
      if ($\exists e_{(b,v)}$ $\forall v \in L1$)
        $L^* \leftarrow (L2 \cap ns_i)$ where $(s_{(b,ns_i)} \in S)$
        findCliques($L1 \cup b, L^*$)

Figure 3.5: **Algorithm 4**: Clique Structure Indexing

of nodes from each complete bipartite $k_{(m,n)}$ and look for cliques that are formed by those nodes. Note that, any clique larger than three nodes in the graph G will be indexed as multiple bipartite structures. Hence, we do not need to consider all nodes in the graph when indexing maximal clique structures. The clique search algorithm works recursively on each node from the $k_{(m,n)}$ as the pivot node in the $L1$ set and considers other nodes as candidate nodes in the $L2$ set. The function, moves each node from the $L2$ set to the $L1$ set if it is connected to all nodes in the $L1$ and then recursively tries to grow the structure with remaining nodes as candidates. When there are no more candidates to consider in $L2$ set then a clique has been identified. Note that, this algorithm is not optimal and better solutions for finding all cliques are proposed in [29, 107].

Finding 4-clique and larger clique structures takes $O(|K|.2^{|k_{m,n}|})$ where $|K|$ is the number of complete bipartite structures, and $|k_{m,n}|$ is the average size of bipartite structures within the graph.

3.4 Evaluation on Internet Topologies

In this section, we use router level Internet topologies to analyze the structural graph indexing approach. In the remainder of the section, we illustrate how our method can be applied to the router level Internet Topology to resolve unresponsive routers.

3.4.1 Graph Transformation

In our experiments, we use iPlane datasets [87]. In these datasets, there are two types of nodes, namely known (i.e., the ones with an IP address) and unknown (i.e., unresponsive). When there is an unresponsive router, it will appear as a '*' in the traceroute output. If multiple path traces pass through an unresponsive router between routers with known IP address, there will be multiple parallel *-substrings between these known nodes.

As an example, in Figure 2.5-a, traceroute queries from e to f will return path traces including *-substrings as $(U, *_1, C)$ and $(U, *_2, C)$ respectively. This may then result in two parallel *-substrings between U and C in the resulting topology map as shown in Figure 2.5-b. In the example, there is a *-substring that includes only one unresponsive router. A similar pattern can be observed for *-substrings of larger lengths in a typical dataset.

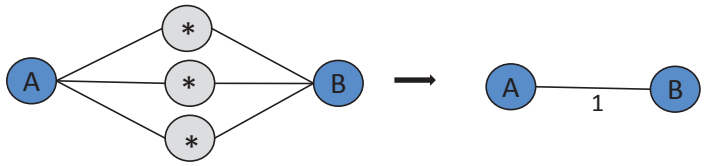


Figure 3.6: Sample Transformation

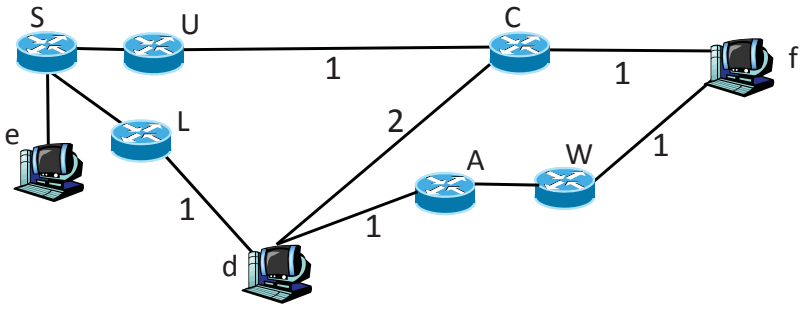


Figure 3.7: Transformation of Graph in Figure 2.5-b

In order to resolve this type of unresponsive routers, we need to detect the same *-substrings (i.e., same length *-substrings with the same known nodes at the end points) [60]. In our method, we ignore all nodes which don't have any unresponsive neighbor, since they do have no effect for the resolution process. While reading the traces from iPlane database, we identify the unresponsive routers which are in between two known nodes. We read all *-substrings from the database and construct a new graph $\bar{G} = (\bar{V}, \bar{E})$. We represent each *-substring as an edge e(a,b,l) where a is the first known node, b is the second known node and l is the label of the edge representing the number of unresponsive nodes between a and b as in Figure 3.6. We add each $e(a, b, l)$ only once to our new graph $\bar{G}$. We add a and b to $\bar{V}$. This process can be called as initial pruning(IP). Figure 3.7 presents the transformed version of the sample topology represented in Figure 2.5-b. After this step, we sequentially index star, complete bipartite, triangle and clique structures in graph $\bar{G}$ with SGI algorithm in order to resolve all unresponsive routers by using the graph based induction technique presented in [60].

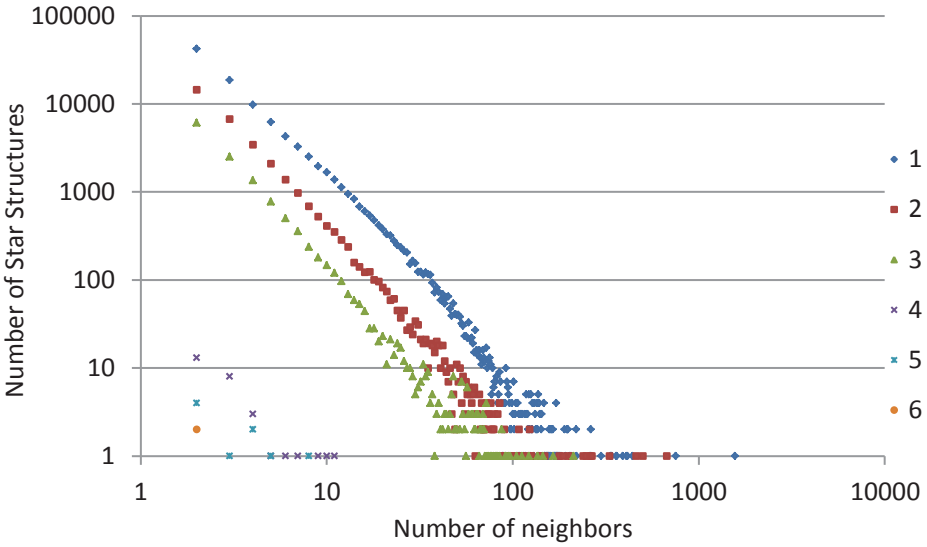
3.4.2 Structure Statistics

We present the indexing results for the datasets collected in 2006, 2007, 2008, 2009, and 2010 in Figures 3.8, 3.9, 3.10, 3.11, and 3.12. In each dataset, there are around 10M unresponsive nodes and 300K known nodes.

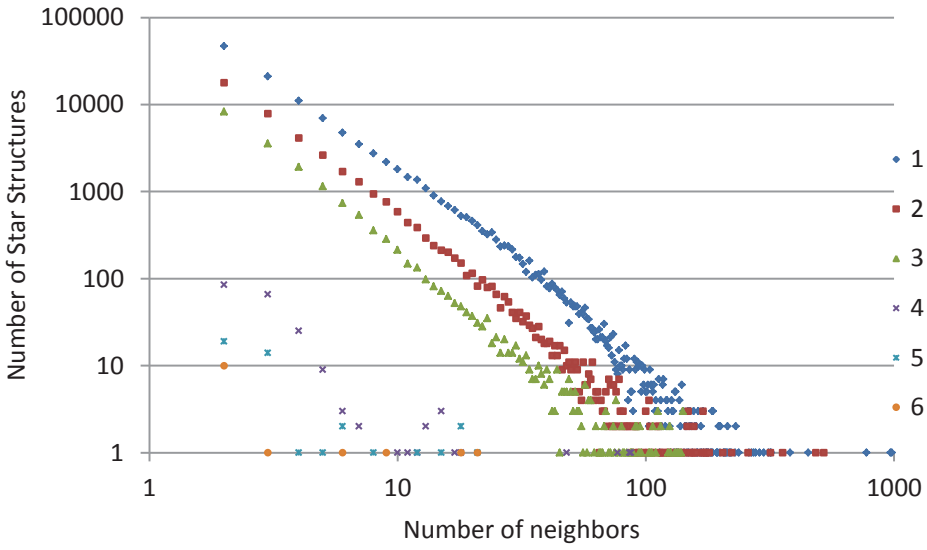
Figure 3.8 represents the star structure distribution for these datasets. In the graphs in this figure, x axis represents the number of leaves of the star structure while y axis represents the corresponding number of star structures for each number of leave. The values in the legend represents the weight (i.e. number of unresponsive routers) of the each edge in the star. According to these graphs, while the number of structures with 1 unresponsive node has not changed considerably in consecutive years, there has been a significant increase in the number of structures with 3 and more unresponsive routers. While there is no star structure with 7 or 8 consecutive unresponsive routers in 2006 and 2007, we observe such structures in following years.

Figure 3.9 represents the bipartite structure distribution of the 2006 dataset, while Figure 3.10 represents the 2010 dataset. In the heatmaps, x axis represents the number of nodes in the left set of the bipartite while y axis represents the number of nodes in the right set. The values in the palette represents the number of the complete bipartites for given left and right set sizes. According to these heatmaps, while the number of set sizes increases the number of bipartite structures found decreases. Additionally, while the edge weight increases the number of complete bipartites observed decrease. However, the number of complete bipartites in the Internet increases year by year. While there is no complete bipartite structure with 4 consecutive unresponsive routers in 2006, we observe such structures in 2010.

Figure 3.11 represents the triangle structure distribution for the years 2006-2010. While the edge weight increases the number of triangles decreases. However,

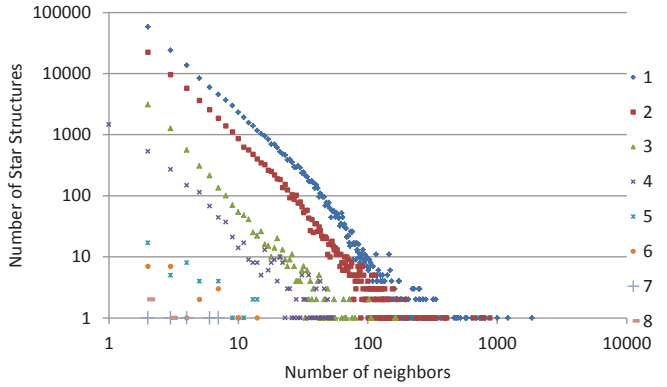


(a) 2006

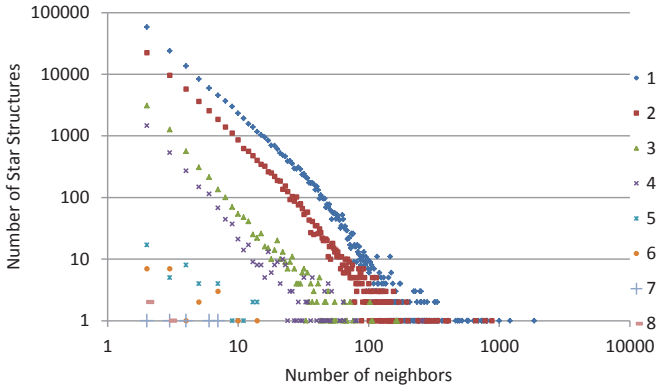


(b) 2007

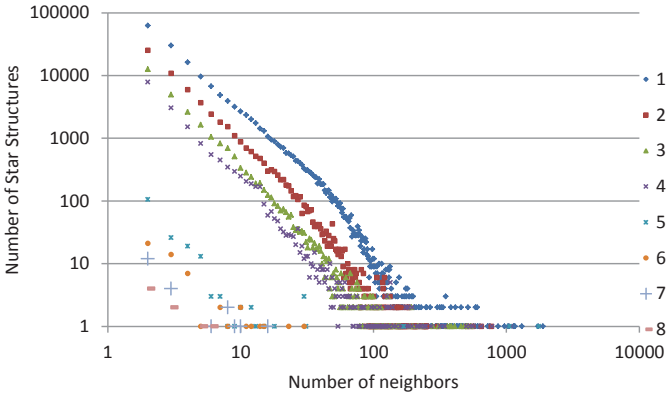
Figure 3.8: Star Structure Distributions



(c) 2008

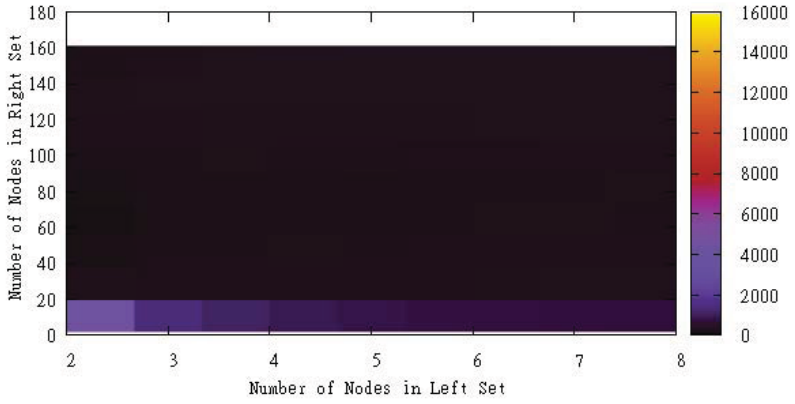


(d) 2009

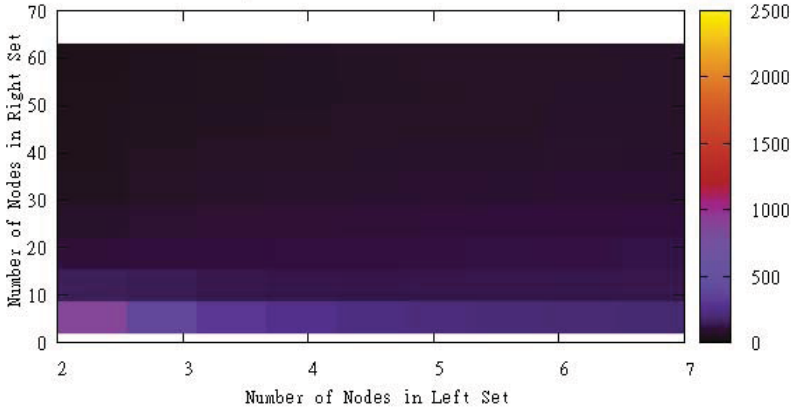


(e) 2010

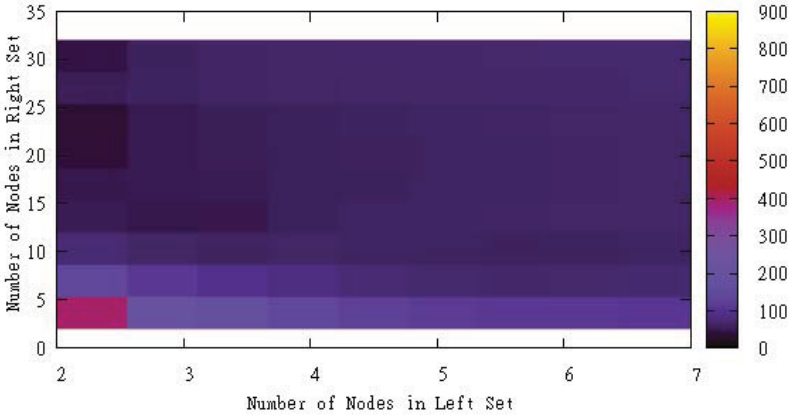
Figure 3.8: Star Structure Distributions



(a) edge weight=1

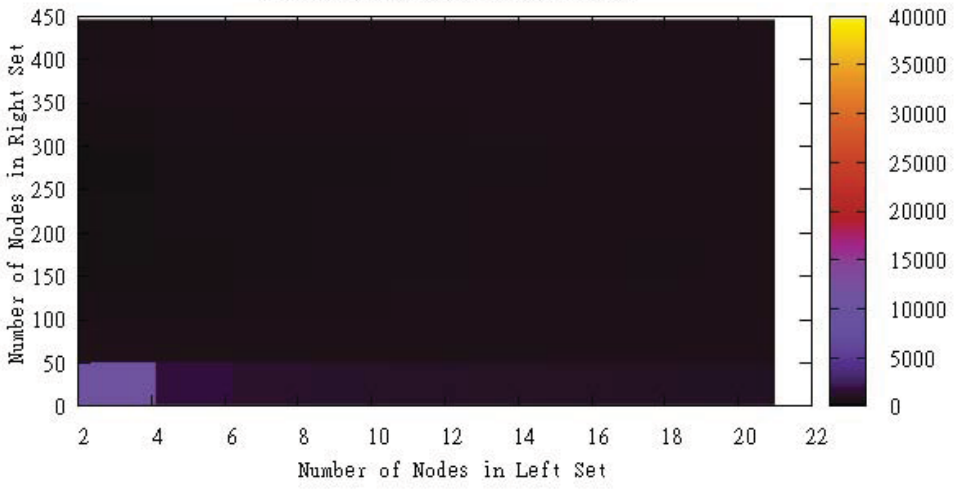


(b) edge weight=2

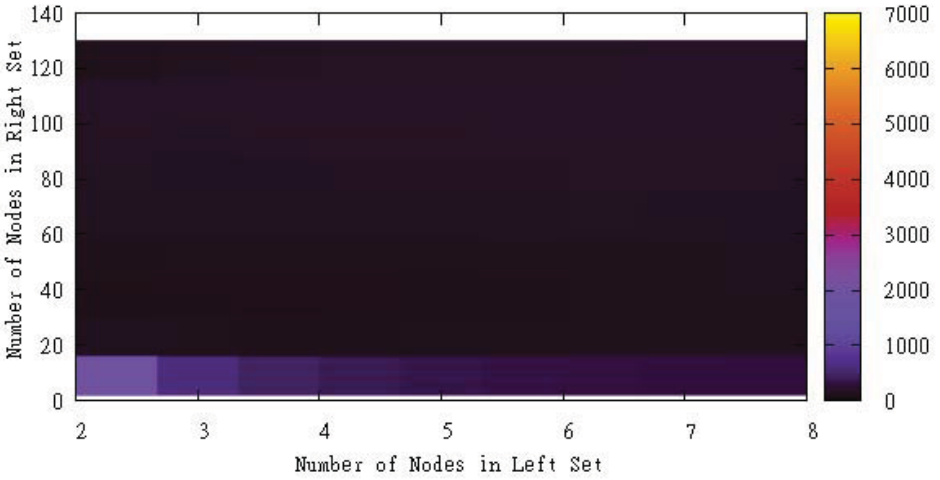


(c) edge weight=3

Figure 3.9: Bipartite Structure Distribution - 2006

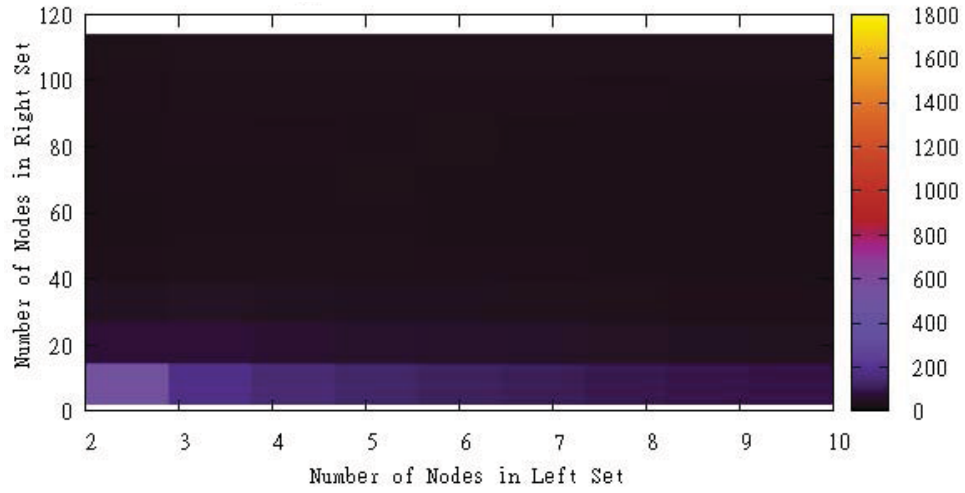


(a) edge weight=1

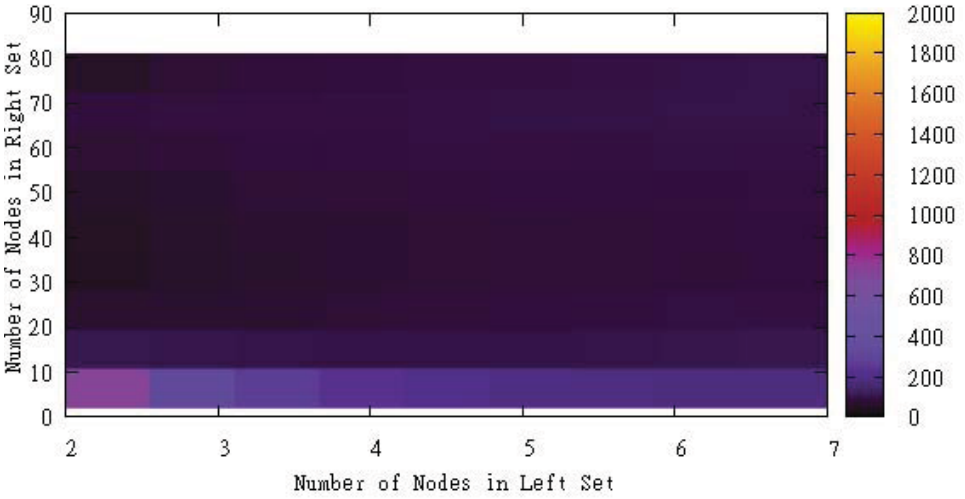


(b) edge weight=2

Figure 3.10: Bipartite Structure Distribution - 2010



(c) edge weight=3



(d) edge weight=4

Figure 3.10: Bipartite Structure Distribution - 2010

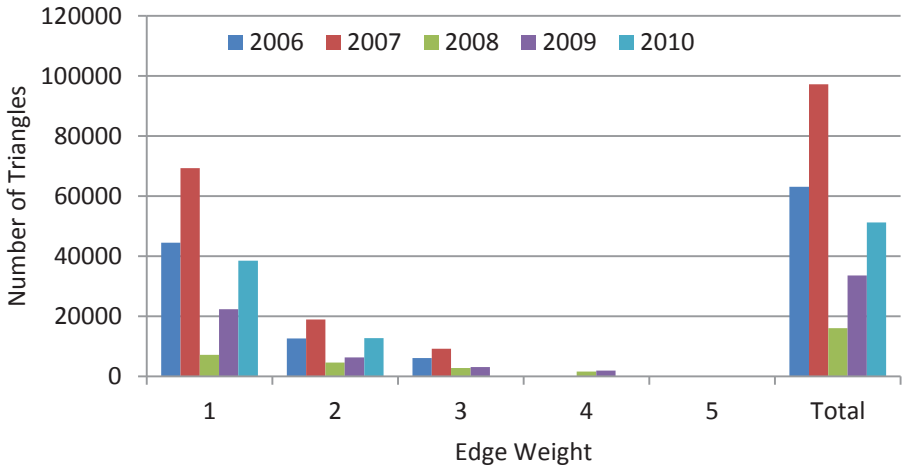
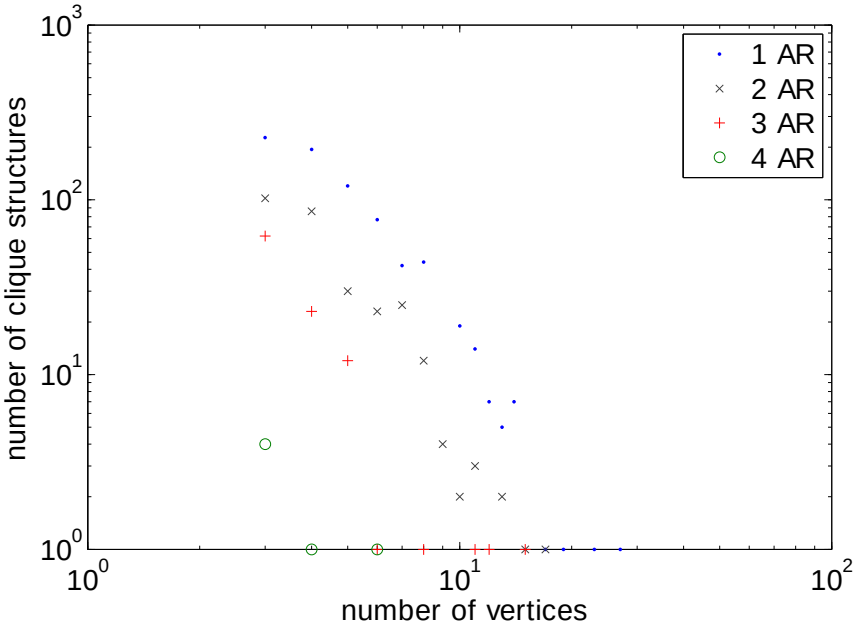


Figure 3.11: Triangle Structure Distribution

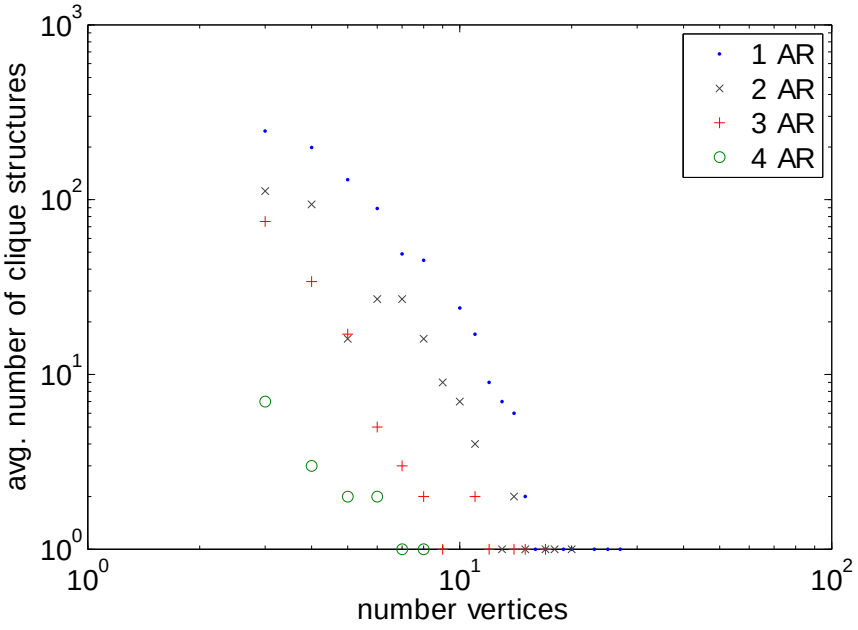
there is no any relation between the number of triangles in the Internet year by year. The highest number of triangles observed in 2007 dataset.

Figure 3.12-(a) represents the clique structure distribution for 2006, while Figure 3.12-(b) represents the average number of cliques observed between 2006-2010. While the edge weight increases the number of cliques decreases. Additionally, the number of cliques in the Internet increases year by year.

All in all, when the length of *-substring increases (i.e. we have more unresponsive nodes between two known node), the number of structures found within the topology decreases. According to our experiments, while the number of structures with 1 unresponsive node has not changed considerably in consecutive years, there has been a significant increase in the number of structures with 3 and more unresponsive routers.



(a) 2006



(b) Average

Figure 3.12: Clique Structure Distribution

Table 3.1: SGI for Resolving unresponsive Routers in iPlane Data-set

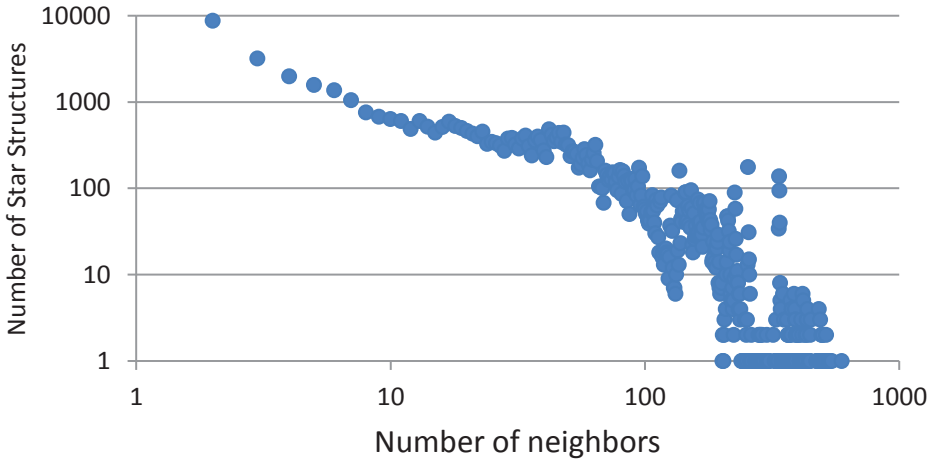
	#Unresponsive nodes	#Resolved
IP	9,063,317	8,269,462
Triangle	793,855	2,864
Bipartite	790,981	172,218
Star	618,763	466,101
Final	152,662	6,940,449

3.4.3 Resolution Results

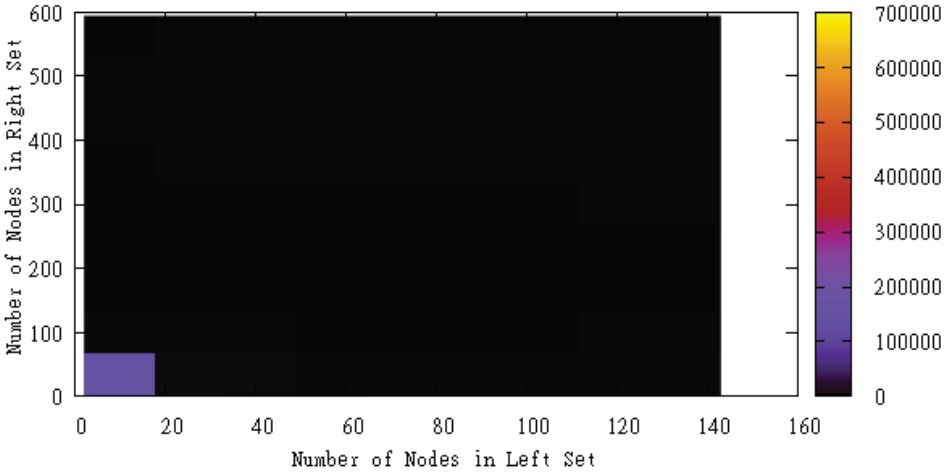
Here, we represent the average resolution results for these datasets. We have around 9M unresponsive nodes in initial data. We first resolve the unresponsive nodes between two known nodes. Then, we use SGI algorithm to resolve unresponsive nodes. Starting from the maximum clique, we resolve all unresponsive nodes within the clique and triangle structures. Then, we resolve unresponsive nodes within the complete bipartite and star structures. The number of resolved unresponsive nodes at each step given in Table 3.1. Using SGI, we resolve more than 98 percent of unresponsive nodes.

3.5 Evaluation on a Wikipedia Graph

In this section, we apply the SGI on a Wikipedia graph. In this graph, Wikipedia pages represented as nodes, and the links between these pages as edges. There are around 100K nodes and 3M edges in this graph. Figure 3.13 shows the star and complete bipartite structure distribution of the Wikipedia graph. When compared to Internet topology datasets above, Wikipedia graph has much more average node degree. Additionally, the number of complete bipartites is higher than Internet topology datasets. SGI found 16.5M triangles in the Wikipedia graph. According to the difference between the number of triangles in Wikipedia and Internet graph, we can conclude that the Wikipedia graph is more clustered than the Internet topology.



(a) Star Distribution



(b) Complete Bipartite Distribution

Figure 3.13: Wikipedia Graph

3.6 Summary

In this chapter, we presented the Structural Graph Indexing (SGI) for efficiently mining complex networks. As indexing feature, we utilize graph structures such as star, complete bipartite, triangle and clique that frequently appear in protein, chemical compound, and Internet graphs. SGI lists all substructures matching structure formulations and other graph structures can be identified and added to the SGI. Different from previous approaches, SGI does not limit the number of nodes in the indexing structure and provides an alternative tool for graph mining algorithms. In evaluation of SGI, we performed experiments on genuine Internet topology datasets to resolve unresponsive routers.

After applying our Structural Graph Indexing (SGI) approach to resolve the unresponsive routers in the Internet topology collected by iPlane [87], we made two main observations. First of all, the number of unresponsive nodes resolved during the clique resolution step is very low despite the high complexity of clique indexing. Hence, we decided to use triangle resolution instead of clique resolution and then we added the triangle indexing and resolution modules to our system. Secondly, we realized that resolving unresponsive routers without identifying the IP alias pairs significantly affects the final graph. Therefore, in Cheleby, we perform the unresponsive router resolution step after identifying the IP alias pairs.

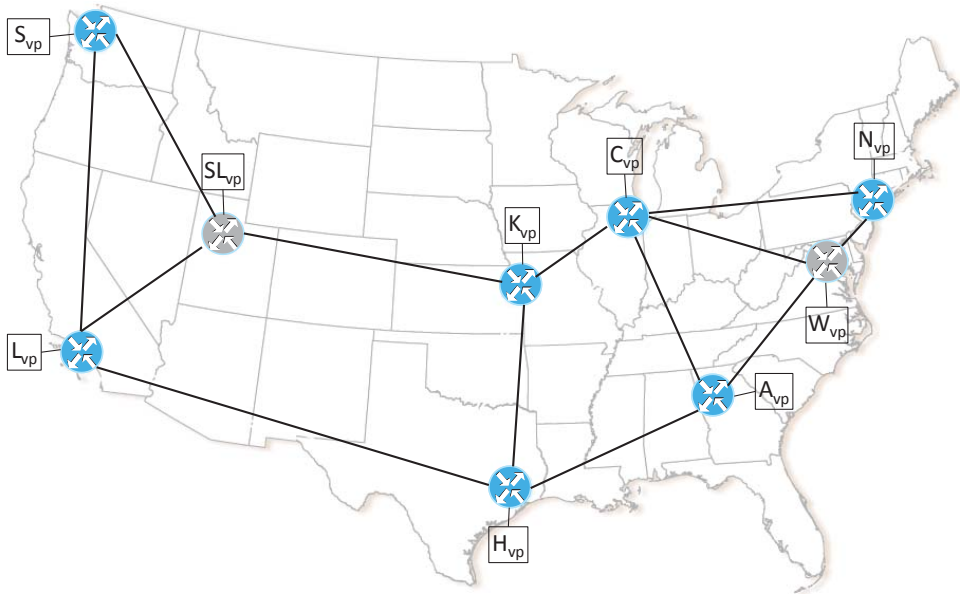
Chapter 4

Unresponsive Routers

Internet topology measurement studies utilize traceroute to collect path traces from the Internet. A router that does not respond to a traceroute probe is referred as an unresponsive router and is represented by a '*' in the traceroute output. Unresponsive router resolution refers to the task of identifying the occurrences of '*'s that belong to the same router in the underlying network. This task is an important step in building representative traceroute-based topology maps and obtaining an optimum solution is shown to be NP-complete. In this chapter[1], we conduct an experimental study to understand how the responsiveness of routers has changed over the last decade and how it differs based on the probing mechanism. We then utilize a novel graph data indexing approach to build an efficient solution to the unresponsive router resolution problem. The results of our experiments significant improvements in accuracy and effectiveness over the existing approaches.

[1]An earlier version of this chapter is currently under review at the IEEE/ACM Transactions on Networking

Hakan Kardes, M.H. Gunes, Kamil Sarac. *Graph Based Induction of Unresponsive Routers in Internet Topologies.*



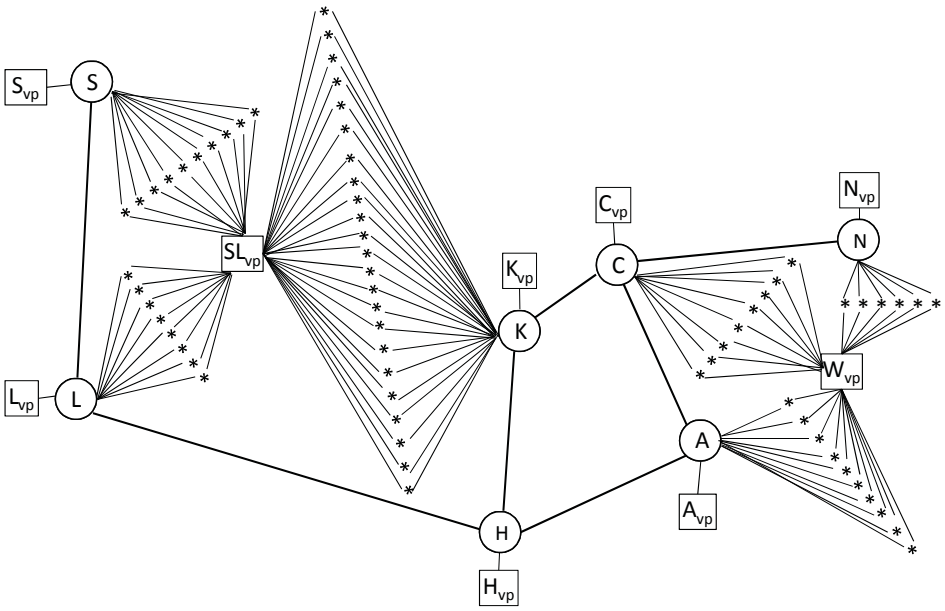
(a) Internet2 Backbone (grey routers are unresponsive)

Figure 4.1: Sample Network

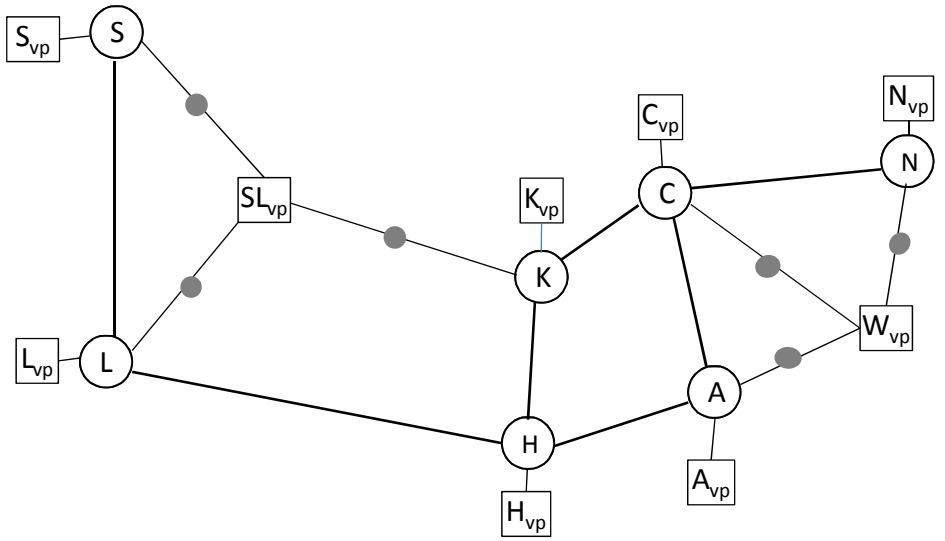
4.1 Introduction

A router that is unresponsive to measurement probes is represented by a '*' in traceroute output and the same router may be probed multiple times in different traces[2]. Without any resolution, each occurrence of '*' needs to be treated as a potentially different router. Depending on the number of unresponsive routers and the topology collection scenario, the collected set of path traces may include a large number of '*'s. For instance, let us consider the Internet2 backbone presented in Figure 4.1-a where squares represent the vantage points attached to each router shown as a circle. Assume that the routers at $Washington, D.C.$ and $SaltLake$ are configured to be unresponsive and path traces are collected between all vantage points. Under these

[2]In this dissertation, we use the term *unknown node* to refer to a '*' in a traceroute output and *unresponsive router* to refer to the actual router that is represented by this unknown node (i.e., by this '*').



(b) Induced Topology (traces between all vantage points)



(c) After Initial Pruning (of unresponsive nodes)

Figure 4.1: Sample Network

assumptions, the topology that is constructed from the 36 path traces between 9 vantage points would be as in Figure 4.1-b (with no resolution). The common approach of pruning (see Sec. 4.3 for details) as in Figure 4.1-c leaves artificial nodes that needs to be further processed.

Internet topology measurement data includes a large number of '*'s. For instance, the iPlane dataset we use in our evaluations in Section 4.2.1 has almost 98% of nodes as unknown (78% after the initial pruning). Hence, timely analysis of large scale data sets, currently tens of millions of traces, requires efficient algorithms. Moreover, the large volume of unresponsive routers in collected Internet graph significantly affects graph characteristics such as degree distribution, and clustering coefficient of the graph (see Sec. 4.4.1). In order to build more representative Internet maps, unresponsive routers should be properly resolved. However, the massive volume of unknown nodes in the collected data set introduces challenges in building efficient solutions.

In this chapter, we first investigate the responsiveness of routers to active network measurements in two directions *(i)* historical router responsiveness to active measurements and *(ii)* current responsiveness to different probe mechanisms. Expanding the earlier work [61] on skitter [94] data sets, we present the prevalence of unresponsive routers in historical Ark [2] and iPlane [87] measurement data sets. For today's practices, we use different types of active probes to observe the responsiveness of routers to them.

We then enhance the *graph based induction* (GBI) approach, based on the earlier work [60], to resolve unresponsive routers in traceroute based topology mapping studies by integrating our novel structural graph indexing [74] approach to efficiently query constructed topology graphs. In our work, we define an induction approach from the practical context of the unresponsive router resolution problem and develop

an efficient implementation which scales to large Internet topology maps with millions of routers. We first analyze the nature of unresponsive routers and identify different types of unresponsiveness. We observe that unresponsiveness due to ICMP rate limiting is an important case that is omitted in previous studies. Then, we visually examine topology maps that are constructed from traceroute data with unresponsive routers. From this visual study, we identify a number of graph structures that are formed among unknown nodes and their known neighbors. We then use *structural graph indexing* to detect these structures in the graph and reduce the unknown nodes (i.e., the occurrences of '*'s) into their corresponding unresponsive routers.

In our evaluations on synthetic topologies, we observe that GBI has better resolution than previously proposed pruning [27] and neighbor matching [46] approaches. Due to their high algorithmic complexity (see Sec. 2.2.3), we did not compare GBI with graph minimization [128], dimensionality reduction [15], and spectral embedding [15] approaches. We also showed that proper resolution of IP Aliases beforehand improves the unresponsive resolution. In addition, we demonstrate the practicality of our approach on genuine datasets collected by *(i)* iPlane with 33M traces, 9M unresponsive and 300K known nodes, *(ii)* Ark with 22.7M traces, 11.4M unresponsive and 1.2M known nodes, and *(iii)* Cheleby [3] with 15M traces, 7.2M unresponsive and 1.2M known nodes.

The rest of this chapter is organized as follows: Section 4.2 presents an experimental study to understand the responsiveness of routers to active probing. Section 4.3 formally defines the unresponsive router resolution problem along with building blocks for the proposed algorithms and introduces our graph based induction approach. Section 4.4 presents our experimental evaluations. Finally, Section 4.5 concludes the chapter.

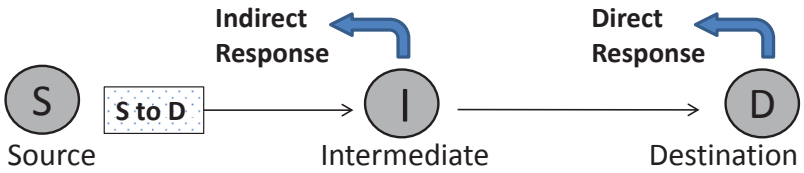


Figure 4.2: Active Probing

4.2 Router Responsiveness Analysis

In this section, we investigate router (un)responsiveness to active probes by enhancing the earlier work [61] which analyzes router responsiveness till 2008. As in Figure 4.2 active probes are divided into two categories as (i) *direct probes* where the destination IP address in the probe packet is the intended destination as in ICMP ping or (ii) *indirect probes* where the destination IP address in the probe packet is some other destination as in traceroute when it probes an intermediate router during a trace.

In both direct and indirect probing, routers might not respond to the probes. Similar to Gunes et al. [61], we classify unresponsive routers into two categories:

- **Permanent unresponsive:** A system may be configured to ignore certain probe packets causing it to be unresponsive with such probing. In addition, a border router may be configured to filter out *(i)* certain types of packets such as unsolicited UDP packets directed to a local host or *(ii)* outgoing ICMP responses originating from nodes within its local domain. Border filtering causes internal nodes to be seen as unresponsive.

- **Temporary unresponsive:** A system may apply ICMP rate limiting and become unresponsive if the rate of the incoming probes exceed a preset limit. Similarly, a system may ignore probe packets when it is congested but respond

to them when it is not. Finally, packets may occasionally be dropped or lost due to routing or overflow. In either case, the router has altering responsiveness.

Note that both can be further classified into two subtypes but an external observer can not differentiate the subtypes. Moreover, a system may have a private (i.e., publicly unroutable) IP address that cannot guarantee node uniqueness but can be marked per AS they originate from.

According to this classification, we mark an unknown node between two known nodes (say, A and B) as temporarily unresponsive when we observe a parallel trace with an IP address (say C) between the same pair of nodes (i.e., A and B). However, if we don't observe any IP address between A and B, we mark it as permanent unresponsive.

4.2.1 Historical Data Analysis

In this section, we use traceroute collected historical data sets to study router reaction to indirect probe messages. We downloaded publicly available historical traceroute data sets from CAIDA (first 10 collected by skitter [94], and the rest collected by Ark [2]), and 6 datasets from iPlane [6]. We utilized data sets that were collected in January of each year by the corresponding infrastructures. CAIDA reports that they had several updates to destination IP address lists and had a major change in their topology collection system in mid 2004 where they utilized dynamic destination lists with increased probing frequency at skitter monitors. In 2008, CAIDA deployed Ark infrastructure to collect the topology data.

First, we look for a trend in the ratio of unresponsive routers in the collected data set in Table 4.1. In the table, *Completed traces* gives the percentage of traces

Table 4.1: Analysis of Historical Responsiveness

		Initial					Unresponsiveness Type	
Dataset	Year	# Vantage Points	# Traces	Completed Traces	# Nodes	Unknown	Permanent	Temporary
Skitter (CAIDA)	1999	5	3.5 M	86.5 %	0.2 M	37.2 %	100 %	0.0 %
	2000	14	14.8 M	83.5 %	0.7 M	45.1 %	100 %	0.0 %
	2001	17	13.4 M	73.6 %	2.1 M	42.2 %	100 %	0.0 %
	2002	20	19.1 M	50.4 %	1.5 M	33.9 %	100 %	0.0 %
	2003	23	24.3 M	54.3 %	1.9 M	29.6 %	100 %	0.0 %
	2004	23	22.9 M	53.0 %	2.4 M	39.1 %	100 %	0.0 %
	2005	22	21,0 M	46.4 %	6.8 M	81.9 %	96.9 %	3.1 %
	2006	19	18.4 M	37.2 %	6.4 M	81.5 %	96.5 %	3,5 %
	2007	18	17.5 M	30.6 %	4.9 M	84.8 %	95.6 %	4.4 %
	2008	11	10.7 M	23.2 %	2.8 M	76.8 %	92.8 %	7.2 %
Ark (CAIDA)	2009	31	22.2 M	7.4 %	12.0 M	86.3 %	96.7 %	3.3 %
	2010	39	24.7 M	8.6 %	10.8 M	85.3 %	96.9 %	3.1 %
	2011	51	22.7 M	9.4 %	12.6 M	87.9 %	96.4 %	3.6 %
iPlane	2006	190	16.6 M	81.8 %	9.8 M	97.5 %	89.9 %	10.1 %
	2007	179	18.8 M	66.8 %	11.7 M	97.4 %	90.1 %	9.9 %
	2008	197	25.7 M	73.2 %	17.8 M	98.1 %	92.4 %	7.6 %
	2009	187	25.2 M	62.1 %	14.6 M	97.9 %	91.9 %	8.1 %
	2010	252	36.7 M	53.5 %	14.5 M	97.7 %	91.0 %	9.0 %
	2011	234	33.4 M	51.1 %	15.5 M	97.8 %	89.7 %	10.3 %

that reached the final destination; *# Nodes* gives the number of nodes within the data set before IP alias and unresponsive router resolutions; and *Unknown* gives the percentage of unknown nodes (where each occurrence of unknown node is counted separately while unique known nodes are counted) in the original data set. The next two columns give the classification of unknown nodes as percentage values.

According to skitter and iPlane data sets, the ratio of path traces reaching their final destination decreases in time. This ratio is very low in traces collected with Ark compared to skitter. The decrease in reachability and the low reachability values are mainly caused by the change in the default ICMP behavior by operating systems, proliferation of firewalls, and the inclusion of a destination from each /24 subnet range that might not correspond to a live system. However, path traces not reaching their final destinations contribute little useful information and considerably slow down the probing process. Hence, CAIDA has developed mechanisms to increase the ratio of traces reaching their final destination.

The ratio of unresponsive routers fluctuates in skitter data but is more stable in Ark data (i.e. around 86%) and iPlane data (i.e. about 98%). These high ratios point to the prevalence of unresponsive router resolution to obtain realistic sample topologies. Another observation from the table is that the ratio of temporarily unresponsive routers increases over the time for all systems in recent years. Yet, they were ignored by previous unresponsive router resolution approaches.

Next, we are interested in the length distribution of path segments formed by consecutive '*'s in path traces. We call a path segment in the form of a $(IP_1, *_1, *_2, \ldots, *_l, IP_2)$ a **-subpath* of length l. We are interested in the frequency distribution of *-subpaths with respect to their length l. Although a *-subpath of length one may have different interpretations about the cause of router unresponsiveness, occurrence

Table 4.2: *-Subpath Characteristics

Dataset	Year	Unique *-subpaths	Same AS	*-subpath length 1	2	3	4	5	>5
Skitter (CAIDA)	2005	225,456	12.6 %	151,133	63,662	6,360	4,301	-	-
	2006	207,067	11.6 %	137,829	59,171	5,828	4,239	-	-
	2007	305,331	14.4 %	212,263	73,263	14,019	5,779	7	-
	2008	231,633	14.0 %	148,182	63,944	13,733	5,772	2	-
Ark (CAIDA)	2009	315,019	30.9 %	194,889	70,728	30,686	18,349	242	128
	2010	370,802	28.6 %	239,212	76,540	34,969	19,971	322	158
	2011	393,570	25.9 %	233,613	87,423	51,261	20,543	411	297
iPlane	2006	472,656	25.6 %	330,672	100,198	40,536	825	269	155
	2007	731,077	25.9 %	510,186	215,924	2,941	1,327	441	242
	2008	931,880	23.3 %	530,698	216,452	110,300	71,097	1,313	1,809
	2009	840,132	23.1 %	500,488	180,283	92,988	64,837	781	728
	2010	844,712	22.7 %	629,468	207,725	3,651	1,781	1,163	833
	2011	851,404	23.0 %	622,900	218,128	4,942	2,579	1,312	1,502

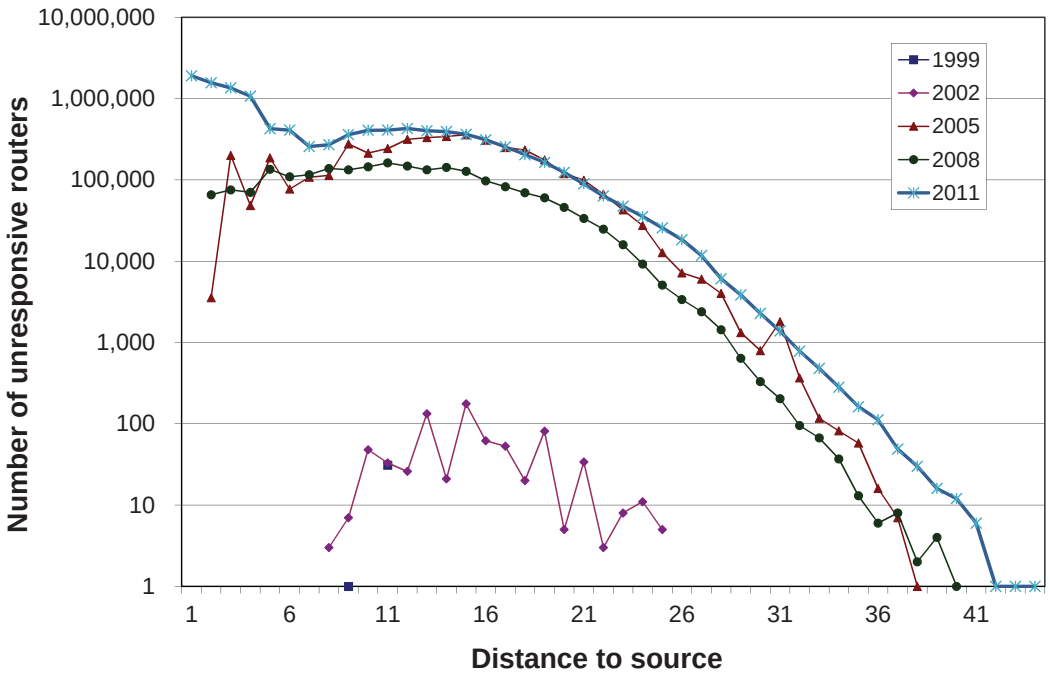
of long *-subpaths may be an indication of ISP policy of preventing active probing in its network.

Table 4.2 represents the *-subpath distribution for CAIDA and iPlane datasets. In the table, we present the number of unique *-subpaths in the original data set. For uniqueness, we represent a *-subpath of length l as a triplet (IP_1, l, IP_2) and avoid counting the duplicate triplets of this form. The data sets prior to 2005 for CAIDA have only length $l = 1$ *-subpaths and are not included in the table. Starting 2005, we observe *-subpaths of longer lengths with the majority of *-subpaths being of length 1 or 2. We partly attribute the behavior of routers to changes in the data collection process such as the increased probing rate. Longer *-subpaths might also be due to growth in networks where more hops of an unresponsive AS are traversed or due to increased use of MPLS tunnels. According to the results for the iPlane data sets, $l = 1$ *-subpaths are almost doubled within the last five years. We observe that both the overall number of unresponsive routers and the longer length *-subpaths have increased as ISPs have become less cooperative to active probing.

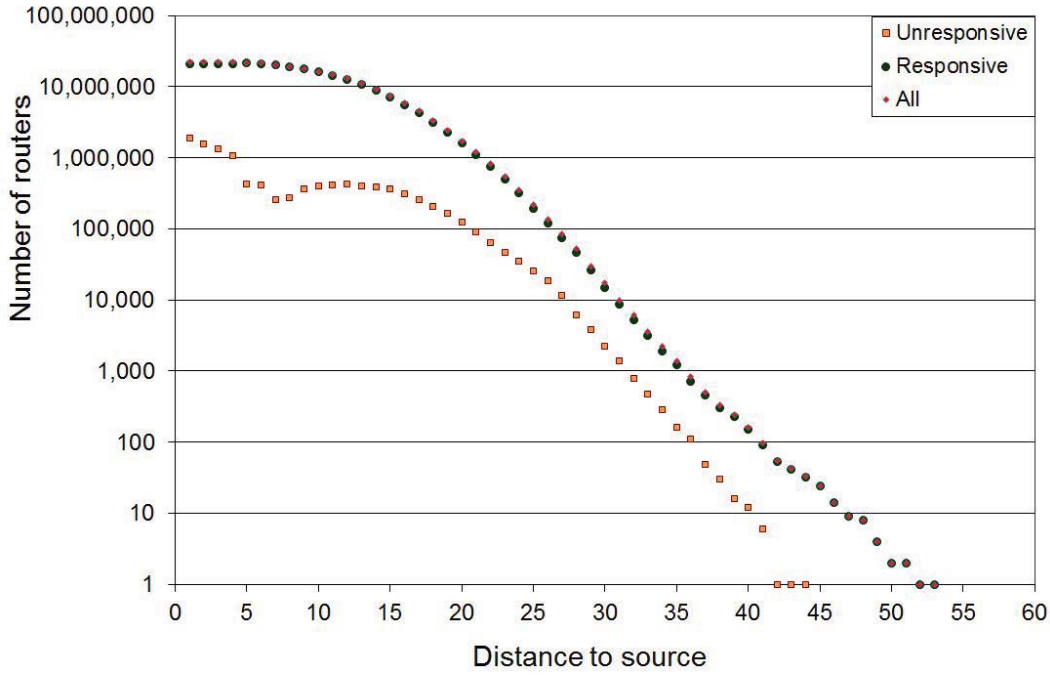
We also look into the percentage of *-subpaths that appear within a single AS in the *Same AS* column. For a given *-subpath, say (IP_1, l, IP_2), we look at the relation between IP_1 and IP_2 and map each such IP address to corresponding AS numbers with AS lookup tool of CYMRU [40]. If the IP addresses share the same AS number, then these IP addresses belong to the same domain and therefore the unresponsive nodes in between most likely belong to the same domain. Given that most *-subpaths are of length 1 or 2 and the probability of two IP addresses being in different ASes is $\tilde{0}.75$ (according to Table 4.2), we think that the majority of *-subpaths originated from routers at domain boundaries or exchange points between neighboring ASes.

Table 4.3: Responsiveness to Direct Probes

Year / Data	Type	All	Router	End-System	.net	.com	.edu	.org	.gov
2008 / Router	ICMP	81.9 %	84.6 %	77.9 %	92.3 %	86.4 %	88.9 %	95.5 %	92.9 %
	TCP	67.3 %	70.4 %	62.8 %	76.7 %	72.6 %	83.2 %	77.3 %	83.0 %
	UDP	59.9 %	64.7%	50.3 %	63.5 %	61.7 %	57.3 %	64.4 %	62.8 %
2011 / WebSite	ICMP	80.4 %	-	80.4 %	84.9 %	86.7 %	53.2 %	83.6 %	37.2 %
	TCP	97.9 %	-	97.9 %	98.3 %	97.8 %	95.8 %	98.2 %	96.9 %
	UDP	46.7 %	-	46.7 %	47.6 %	50.9 %	21.0 %	45.8 %	14.4 %



(a) Unresponsive Routers



(b) Routers for 2011 CAIDA data set

Figure 4.3: Historic Distance Distribution

We are also interested in the position of unresponsive nodes within path traces and counted the number of unknown nodes at each hop distance from the vantage point. Figure 4.3-a presents the distance distribution of unknown nodes for five different years of CAIDA data sets. According to the figure, early data sets (i.e., before 2005) contain small number of unknown nodes that were mostly distributed 10 to 20 hops away from the source. On the other hand, recent data sets include much more unknown nodes majority of which appear 3 to 25 hops away from the source. The figure also shows a high number of unknown nodes at a distance of 2 from the source for the 2011 data set. A close examination of the corresponding data set shows that this is due to the existence of a permanently unresponsive router at a 2 hops distance to one of the vantage points. Figure 4.3-b presents the distance distribution for known, unknown, and all of the nodes for the 2011 CAIDA data set. Since the number of unknown nodes is relatively smaller than the number of known nodes when accounting for each occurrence separately, the distance distribution of all the nodes is quite similar to distance distribution of known nodes. Moreover, unlike the distribution of known nodes, there is a decrease in the number of unknown nodes between distance 1 to 9. Starting from hop distance 10, they start to exhibit a similar trend.

4.2.2 Responsiveness to Different Probe Mechanisms

In this section, we observe router responsiveness to direct and indirect probe messages (see Figure 4.2 for distinction). Destination responses may change with different probe mechanisms and hence the resulting topologies under each probing approach may vary. For the analysis in this section, we also include the analysis done by Gunes et al. in 2008 [61] to compare the difference between 2008 and 2011. To analyze

responsiveness to these different protocols, we first issued UDP, TCP, and ICMP based direct probes to several IP addresses and recorded the response (or the lack of it). For each case, we issued three probes from a host and expected to receive at least one response to consider the probed node as responsive. In general, UDP probes are expected to elicit *ICMP Port Unreachable* messages; TCP probes (*TCP SYN* packets) are expected to elicit *TCP SYNACK* or *TCP RST* messages; and ICMP probes (*Echo Requests*) are expected to elicit *ICMP Echo Reply* messages. Finally, we issued another set of ICMP probes with *IP Route Record* option set but observed a very small response rate and excluded them from the discussion.

Table 4.3 presents the response rate that we observed during our direct probing in 2008 and 2011. *All* column indicates the response rate of all IP addresses (where each IP is separately counted as whether responsive, i.e., known, or not, i.e., unknown). We then group these IP addresses as belonging to *routers* or *end systems* by analyzing whether they appear within path traces. The heuristic approach classifies an IP address as a router if it appears within any path trace, and as end-system otherwise [3]. We also classify end systems based on their top level domain extensions (only 5 of them are presented) in the following columns.

Initially, we used 537K IP addresses obtained from skitter and iPlane, in 2008 [61]. These IP addresses were collected by running traceroute queries, i.e., they belong to routers/systems that responded to indirect/direct probe messages from skitter and/or iPlane systems. However, during our TCP based active probing, we received several complaints from a national ISP indicating that our probe messages raised alerts in their monitoring system.

[3]This approach may incorrectly mark a router as an end-system when it only appears as the last hop of traces

To minimize such complaints, in 2011, we used 537K new IP addresses of the top visited web sites from `http://alexa.com/topsites`.

According to the 2008 router data, ICMP based direct probes had the highest rate of responses followed by TCP and then UDP. The results suggest that TCP based probes were more welcomed than the UDP ones. However, in 2011, TCP based direct probes have the highest response rate followed by ICMP and then UDP probes. This is skewed because we used IP addresses of the web sites which are in general responsive to TCP probes. In both years, UDP based probes have the lowest response rate while it decreased considerably from 60% in 2008 to 47% in 2011. The response ratio of ICMP based probes is similar for both years (i.e., 81.9% for 2008 and 80.4% for 2011). The 2008 router IP data indicates that many routers that respond to indirect query probes, i.e. traceroute, do not respond to direct query probes. Moreover, in 2011 data, we observe that end systems typically do not respond to UDP probes.

In the next step, we issued DNS queries to obtain the host names corresponding to these IP addresses. We use the host name extensions to classify our IP addresses into several top level domains (TLDs) including .gov, .net, .org, .edu, and .com and look at the responsiveness of each group. Our data set had host names with other TLDs, e.g., .jp, .fr, .info, which we did not include in the table as they were a smaller sample and had similar responsiveness. In 2008, the responsiveness of routers in different TLDs are similar. However, in 2011, ICMP and UDP probes to the .edu and the .gov extension end systems have very low response rate, while the TCP based probes have a similar pattern across TLDs.

In the final step, we issued ICMP, TCP, and UDP based traces toward 306K and 537K of IP addresses in 2008 and 2011, respectively, to observe router responsiveness to different indirect probes in Table 4.4. Similar to Table 4.1, we counted

Table 4.4: Responsiveness to Indirect Probes

Type	Year	#Traces	Completed	#IPs	#Nodes	Unres.
ICMP	2008	306 K	93.1 %	313 K	1.0 M	68.7 %
	2011	537 K	88.8 %	770 K	2.0 M	66.5 %
TCP	2008	306 K	73.4 %	277 K	1.0 M	72.3 %
	2011	537 K	96.0 %	697 K	1.1 M	36.6 %
UDP	2008	306 K	45.0 %	210 K	1.5 M	86.0 %
	2011	537 K	64.4 %	201 K	1.5 M	86.6 %

each occurrence of unknown node separately while only counted unique known nodes. In 2008, over 93% of our ICMP based probes reached their final destination whereas the ratios for TCP and UDP based probes were around 73% and 45%, respectively. These results suggest that most network operators were more cooperative with ICMP based queries while more than half of the operators blocked UDP based queries in their domain. However, in 2011, about 96% of our TCP based probes reached their final destination while the ratios for ICMP and UDP based probes were around 89% and 65%, respectively. Note that the results for 2011 is inflated for TCP since the web sites are normally responsive to TCP packets. Additionally, unresponsiveness ratios for ICMP and UDP based traceroute are similar for both years with UDP being the least effective to elicit responses. This is also reflected in the topology size, i.e., *#Nodes*.

4.2.3 Summary

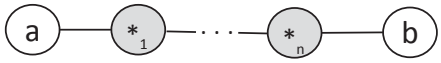
In this section, we presented an experimental study to understand the responsiveness of routers to active probing. In our historical analysis, we observed that responsiveness reduced during the last decade. We also observed that network operators are increasingly rate limiting probe responses. Another observation from our study is that the destination reachability considerably reduced over the time indicating that

systems (i.e., routers and end systems) are increasingly unwilling to respond to direct probes. We also observed that routers are less willing to respond to direct active probes as compared to indirect active probes. Finally, our experiments show that the responsiveness of routers changes with the type of the probes; ICMP based probes eliciting the highest response rate and UDP based ones eliciting the lowest. Even though TCP based probes receive responses much better than UDP based ones, this type of probes are more likely to raise security alerts.

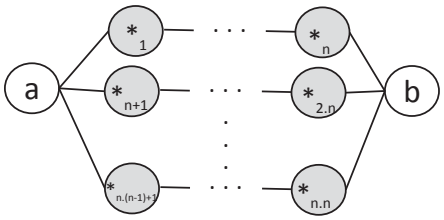
Based on the findings in the measurement study, in our unresponsive router resolution approach, we addressed the lack of (i) *temporary unresponsive router resolution* as the ratio of temporarily unresponsive routers have been increasing and (ii) **-subpath resolution* as the length of *-subpaths and the number of longer ones have been increasing. Neither of these cases are properly addressed by previous resolution studies while both appear increasingly often in the recent years.

4.3 Graph Based Induction for Unresponsive Router Resolution

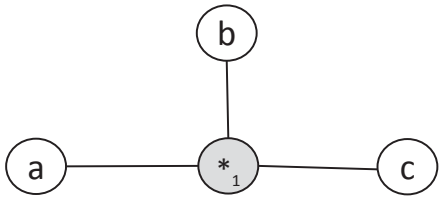
In this section, we present *graph based induction* (GBI) technique to resolve unresponsive routers that introduce a large number of artificial nodes in traceroute-based topology maps [60]. We first formulate a number of graph structures that are observed in traceroute-based sample topologies. First of all, the number of unresponsive nodes resolved during the clique resolution step is very low despite the high complexity of clique indexing. Hence, we decided to use triangle resolution instead of clique resolution and then we added the triangle indexing and resolution modules to our system. These structures are shown in Figure 4.4-a,-c,-e,-g and the corresponding



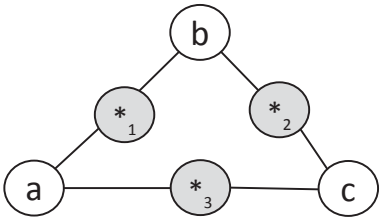
(a) Real topology



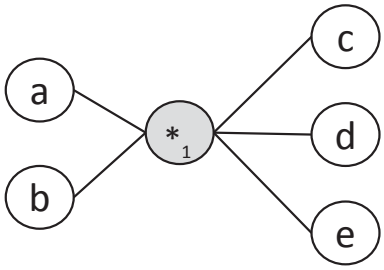
(b) Observed parallel topology



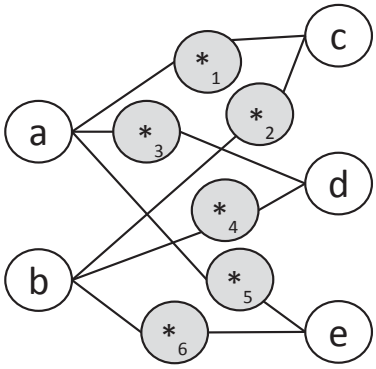
(c) Real topology



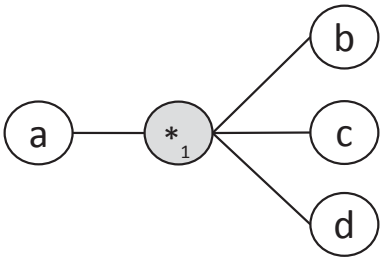
(d) Observed triangle-like topology



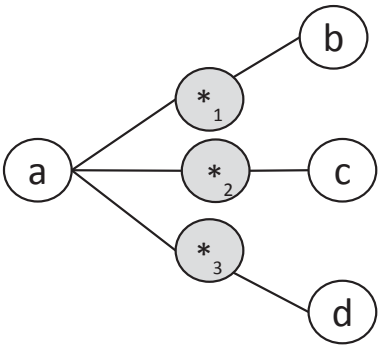
(e) Real topology



(f) Observed complete bipartite-like topology



(g) Real topology



(h) Observed star topology

Figure 4.4: Structures (genuine and observed)

connectivity relations in the underlying actual network are shown in Figure 4.4-b,-d,-f,-h, respectively. For instance, when there are m traces through a sequence of n unresponsive routers as in Figure 4.4-b, we observe m parallel unknown sequences as in Figure 4.4-a. Similarly, when there are traces between all neighbors of an unresponsive router as in Figure 4.4-d, we observe a clique structure as in Figure 4.4-c after parallel sequences are resolved. Figure 4.4-e appears when there are traces from one set, i.e., {a,b}, to the other set, i.e., {c,d,e}, of bipartite graph in Figure 4.4-f. Finally, Figure 4.4-g will appear if there are traces from a source node to multiple destinations in Figure 4.4-h.

Multiple network topologies with unresponsive routers can result in the same observed topology. Yao et al. analyzes this issue and proposes to use the minimal topology under trace and distance preservation conditions as the underlying topology for an observed network topology [128]. We, however, assume that the minimal topology that satisfies the *trace preservation* condition is the underlying topology for observed structures in collected Internet topology datasets. In order to solve the unresponsive router problem, we replace the subgraphs Figures 4.4-a,-c,-e,-g with the subgraphs in Figures 4.4-b,-d,-f,-h within collected topology datasets. Note that networks in Figures 4.4-b,-d,-f,-h are the minimum possible underlying networks for the sampled networks shown in Figures 4.4-a,-c,-e,-g, respectively.

We modify our novel *structural graph indexing* technique [74] where we search for structures similar to the identified ones in traceroute-collected topology data and transform unknown nodes in them into their corresponding routers with *graph based induction* approach. In this process, we first start with a router-level topology graph $G = (V, E)$ that is constructed from a set of traceroute-collected path traces. We find all *-subpaths $u^*_{(a,b,l,A)}$ in this graph, and then apply *structural graph indexing* to

efficiently perform queries and *graph based induction* to resolve unresponsive routers based on the structures shown in Figure 4.4.

In graph indexing, we index predefined structures in a given network so that subsequent queries regarding the network graph are more efficient. In our case, we index star, complete bipartite, and triangle structures in a given graph. An important difference of our approach from the previous indexing studies is that we do not limit the size of candidate subgraphs. We try to index maximal graphs that match the structure formulation. For instance, a maximal complete-bipartite is a complete-bipartite that cannot be extended by adding one more vertex from the graph. In order to reduce computational complexities, we index the structures within the original graph in a consecutive manner. That is, we first identify star structures, then complete-bipartite, and finally triangle structures from the preceding ones as detailed below.

4.3.1 Structural Graph Indexing

In this section, we present each of the identified structures, their underlying topology, and the structural graph indexing of these structures for graph based induction.

4.3.1.1 Parallel *-subpath Structures

First common pattern that we observe in traceroute-based topology maps is the occurrences of same length *-subpaths with the same known nodes at the ends of each *-subpath. While collecting path traces from a vantage point, an unresponsive router may appear as '*' in multiple path traces resulting in multiple parallel *-subpaths between the same known nodes. As an example in Figure 4.1-a, traceroute queries

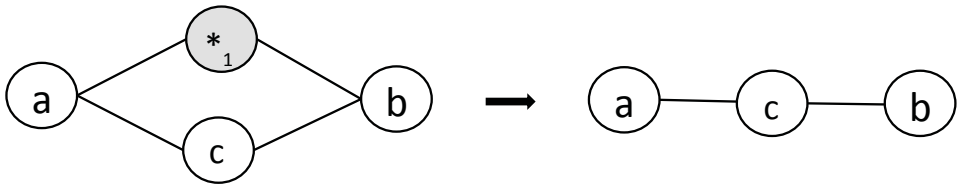
Let $G = (V, E)$; $V \leftarrow \emptyset$; $E \leftarrow \emptyset$; $U \leftarrow \emptyset$; $maxLength \leftarrow 0$;
for (each $trace$ in $\bigcup trace(v_i, v_j)$)
  $V \leftarrow V \cup \{a, b\}$; $E \leftarrow E \cup \{e_{(a,b)}\}$ $\forall u_{(a,b,0,\emptyset)} \in trace$
  for (each $u^*_{(a,b,l,\{*_1,*_2,...,*_l\})} \in trace$)
    if $(\neg\exists\ u^*_{(a,b,l,\_)} \in U)$
      $V \leftarrow V \cup \{a, *_1, *_2, ..., *_l, b\}$
      $U \leftarrow U \cup u^*_{(a,b,l,\{*_1,*_2,...,*_l\})}$
      $E \leftarrow E \cup \{e_{(a,*_1)}, e_{(*_l,b)}\} \cup \{e_{(*_i,*_{i+1})} \forall i, 1 \leq i < l\}$
      if $maxLength < l$
        $maxLength \leftarrow l$

Figure 4.5: **Algorithm 5**: Finding Parallel *-Subpaths

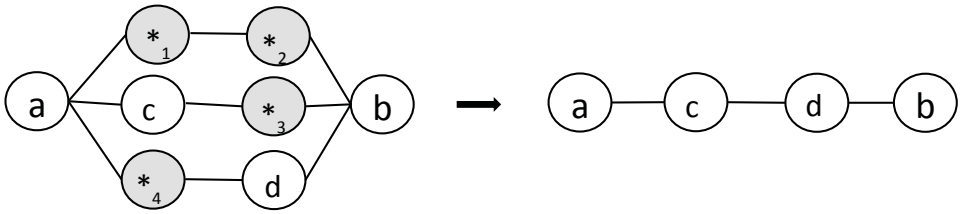
from A_{vp} to W_{vp} return path traces including *-subpaths such as $(A, *_1, W_{vp})$. Similarly, traceroutes from W_{vp} to A_{vp} result in additional *-subpaths such as $(W_{vp}, *_2, A)$. When path traces between all vantage points are collected, we observe 10 parallel *-subpaths between A and W_{vp} in the resulting topology map as shown in Figure 4.1-b. Note that in this example *-subpaths include only one unresponsive router. A similar pattern can be observed for *-subpaths of longer lengths as in Figure 4.4-a.

Resolution of unresponsive routers in this type of structures requires detection of similar *-subpaths (i.e., same length *-subpaths with the same known nodes at their end points). The *Algorithm 1* in Figure 4.5 provides a graph search module for parallel *-subpaths. In the algorithm, we extract all *-subpaths $u^*_{(a,b,l,\{*_1,*_2,...,*_l\})}$ from the path traces and identify the same length *-subpaths with the same known end nodes (i.e., a and b) to merge unknown nodes with each other. The algorithm also builds the initial data structures that we will utilize in the graph indexing to resolve other structures.

While reading the traces, each $u^*_{(a,b,l,N)}$ is stored based on the known end nodes a and b in a hash table. Subsequently read $u^*_{(c,d,k,P)}$'s are then compared to the ones



(a) Resolution of length 1 *-subpath



(b) Resolution of longer *-subpaths

Figure 4.6: Resolution of Temporarily Unresponsive Routers

with the same hash value in the data structure. This results in a complexity of $O(|U|.log(|U|))$ where $|U|$ is the number of *-subpaths.

Another related pattern is caused by routers that apply ICMP rate limiting or that stay unresponsive when congested, i.e., temporary unresponsiveness. Such a router may appear as a known node in some path traces and may appear as a '*' in others. For instance, in Figure 4.6-a, an ICMP rate limiting router c may cause occurrences of related subpaths in the form of $(\ldots, a, c, b, \ldots)$ and $(\ldots, a, *_1, b, \ldots)$ in different traces, and we resolve $*_1$ to c. However, in some cases, there might be a subpath $(\ldots, a, d, b, \ldots)$ as well. In such cases, we resolve $*_1$ to either c or d only if one of them is marked as temporarily unresponsive. Additionally, as in Figure 4.6-b, ICMP rate limiting routers c and d may cause occurrences of related subpaths in the form of $(\ldots, a, *_1, *_2, b, \ldots)$, $(\ldots, a, c, *_3, b, \ldots)$ and $(\ldots, a, *_4, d, b, \ldots)$ in different traceroute outputs. In this case, we resolve $*_1$ and $*_4$ to c; and $*_2$ and $*_3$ to d.

The *Algorithm 6* in Figure 4.7 resolves the temporarily unresponsive routers. In the algorithm, for each *-subpath $u^*_{(a,b,l,\{*_1,*_2,\ldots,*_l\})}$, we look for a same length l

INPUT: $G = (V, E)$ and U from Algorithm 5;
for (each $u^*_{(a,b,l,\{*_1,*_2,...,*_l\})} \in U$) where $l <= 5$
  if ($\exists\ u_{(a,b,l,\{v_1,v_2,...,v_l\})}$)
    set $*_1 \leftarrow v_1$, $*_2 \leftarrow v_2$, ..., $*_l \leftarrow v_l$ in G
    $U \leftarrow U - u^*_{(a,b,l,A)}$

Figure 4.7: **Algorithm 6**: Resolving Temporarily Unresponsive Routers

subpaths between the same end nodes a and b. Finding subpaths for given end nodes in a naive manner might result in a high time complexity. While building the initial graph in algorithm *Algorithm 5* in Figure 4.5, we also index the neighbors of each node in our node structure. In order to find the subpaths for given end nodes, we find $\lceil l/2 \rceil$-hop neighbor set of the left end node and $\lfloor l/2 \rfloor$-hop neighbor set of the right end node. If the intersection set of these two sets is not an empty set, we identify the subpath between the end nodes. The algorithm takes $O(|U|.(a_{nd})^2)$ time where $|U|$ is the number of *-subpaths and a_{nd} is the average node degree of all nodes. Note that we limit the length of *-subpaths to be processed in this step by 5 as longer *-subpaths increase the time complexity and the number of such subpaths are relatively small according to data sets in Table 4.2.

4.3.1.2 Star Structures

After building *-subpath database from path traces, we build an index of the star structures (e.g., Figure 4.4-g) that will be utilized in resolving unresponsive routers as in Figure 4.4-h and finding bipartite and triangle substructures. The star structure typically appears in path traces collected from a single vantage point toward a number of destinations or from multiple vantage points toward the same destination. The observed topology looks like the one presented in Figure 4.4-g. We identify this type

INPUT: $G = (V, E)$ and U from Algorithm 6;
$maxLength$ from Algorithm 5; $S \leftarrow \emptyset$;
for (each *node* $v \in V$)
  for (i=1:maxLength)
    $S \leftarrow S \cup s_{(v,i,\emptyset,\emptyset)}$
for (each $u^*_{(a,b,l,A)} \in U$)
  $s_{(a,l,N,M)} \leftarrow s_{(a,l,(N\cup\{b\}),(M\cup A))}$
  $s_{(b,l,N,M)} \leftarrow s_{(b,l,(N\cup\{a\}),(M\cup A))}$
for (each $s_{(v,l,N,M)} \in S$)
  if $|N| < 2$
    $S \leftarrow S - s_{(v,l,N,M)}$

Figure 4.8: **Algorithm 7** - Star Structure Indexing

of structures by clustering unresponsive neighbors (e.g., $*_v$) of nodes (e.g., a, the head node of *-subpaths in Figure 4.4-g).

We index maximal star structures for each node $v_i \in V$ using *Algorithm 7* in Figure 4.8. Star structures within a graph $G = (V, E)$ are represented as $s_{(v_i,l,N_i,M_i)}$ where v_i is the pivot node, l is the number of unknown nodes between the v_i and each of its l-neighbors, N_i is the set of all l-neighbors of v_i, and M_i is the set of all unknown nodes in this star structure. The algorithm builds on the *-subpaths database U and produces star structure database S.

The algorithm first builds a star structure $s_{(v_i,0,\emptyset,\emptyset)}$ for each known node $v_i \in V$. Then, for each *-subpath $u^*_{(a,b,l,A)}$, it appends node b to the neighbor set of the star structure of node a with length l, and vice versa. It also appends the set A to M set of both nodes' star structures. After processing all *-subpaths, the algorithm removes star structures $s_{(v,l,N,M)}$ that have less than two neighbors, i.e. $|N| < 2$. The overall run time complexity of the algorithm is $O(|V| + |U|)$.

INPUT: $G = (V, E)$ from Algorithm 6; S from Algorithm 7;
$K \leftarrow \emptyset$;
for (each $s_{(a,l,N,M)} \in S$)
  $L_{can} \leftarrow \emptyset$
  for (each $b_i \in N$)
    $L_{can} \leftarrow L_{can} \cup N^*$ where ($\exists\ s_{(b_i,l,N^*,M^*)} \in S$)
  $L_{can} \leftarrow L_{can} - \{a\}$
  $R_{can} \leftarrow N$
  for (each $v_i \in L_{can}$)
    $R_{new} \leftarrow R_{can} \cap N_i^+$ where ($\exists\ s_{(v_i,l,N^+,M^+)} \in S$)
    if ($|R_{new}| \geq 2$)
      $L_{new} \leftarrow \{a\} \cup \{v_i\}$
      for (each $v_j \in L_{can}$)
        if ($R_{new} \subset N_j^{\#}$) where ($\exists\ s_{(v_j,l,N^{\#},M^{\#})} \in S$)
          $L_{new} \leftarrow L_{new} \cup \{v_j\}$
          $M_{new} \leftarrow M_{new} \cup M^{\#}$
      $K \leftarrow K \cup k_{(L_{new},R_{new},l,M_{new})}$

Figure 4.9: **Algorithm 8** - Complete Bipartite Structure Indexing

4.3.1.3 Complete Bipartite Structures

After building Star structure database S, we index complete bipartite structures K. A complete bipartite subgraph (e.g, a 2x3 complete bipartite in Figure 4.4-e) is formed among the known neighbors of an unresponsive router (i.e, $*_1$ for Figure 4.4-f). This type of structure occurs when path traces are between two sets of the unresponsive nodes' neighbors (i.e., $\{a, b\}$ and $\{c, d, e\}$ for Figure 4.4-e). In this case, topology database includes *-subpaths $(n_i, *_v, n_j)$ among all known neighbors of $*_v$. In general, this structure frequently occurs in paths collected using (k,m)-traces (i.e., tracing from a relatively small number of k vantage points to a larger number of m destinations) that is very common in topology mapping studies.

We index all complete bipartite structures using *Algorithm 8* in Figure 4.9. We represent a complete bipartite graph as $k_{(V_1,V_2,l,M)}$ where V_1 and V_2 are the two disjoint sets of neighboring nodes, l is the number of unknown nodes between the nodes v_i and v_j for any two nodes $v_i \in V_1$ and $v_j \in V_2$, and M_i is the set of all unknown nodes in this complete bipartite structure. The algorithm builds on the star database S from *Algorithm 7* in Figure 4.8.

In the algorithm, for each star structure $s_{(a,l,N,M)}$, we identify the maximal complete bipartite involving the node a. For this purpose, we first identify two candidate sets of nodes which will constitute the left and right hand side of the bipartite structure involving the node a. R_{can} set represents the candidates for the right side of the complete bipartite and is also the neighbor set N of this star structure. L_{can} set indicates candidates for the left side of the complete bipartite. This set consists of all l-neighbors of each node in N. We first find a $k_{(V_1^*,V_2,N^*,M^*)}$ where $m^* = 2$ and then grow it to $k_{(V_1,V_2,N,M)}$ where $m \geq 2$. Finding $k_{(V_1^*,V_2,N^*,M^*)}$, we iterate over each candidate node in the L_{can} as a pivot node and determine its neighbor intersection with that of the node a. If the intersection set is larger than two, these nodes belong to the right hand side. After determining the nodes in the R_{can} set, we grow the left hand side (i.e., L_{can}) and hence the $k_{(V_1^*,V_2,N^*,M^*)}$ structure by finding all nodes that has the right hand side nodes (i.e., R_{new}) as a neighbor. Overall, finding complete bipartite graphs takes $O(|S|.a_N^2)$ where $|S|$ is the number of star structures in the graph, and a_N is the average neighbor set size of all stars.

4.3.1.4 Triangle Structure

Finally, we build indexing of the triangle structures. A triangle is formed between an unresponsive router $*_v$ and its known neighbors $\{n_1, n_2, n_3\}$ when we consider the

INPUT: $G = (V, E)$ from Algorithm 6; S from Algorithm 7;
$T \leftarrow \emptyset$; $I \leftarrow \emptyset$;
for (each $s_{(a,1,N,M)} \in S$)
  for (each $N_i \in N$) where $N_i > a$
    if ($\exists$ $s_{(N_i,1,P,M^i)}$)
      $I \leftarrow N \cap P$
      for (each $I_i \in I$) where $I_i > N_i$
        $T \leftarrow T \cup t_{(\{a,N_i,I_i\},1,\{A^1,A^2,A^3\})}$
        where $u^*_{(a,N_i,1,A^1)}$, $u^*_{(a,I_i,1,A^2)}$, $u^*_{(N_i,I_i,1,A^3)}$

Figure 4.10: **Algorithm 9** - Triangle Structure Indexing

known neighbors of $*_v$. This type of structure occurs when path traces exists between all three known neighbors of an unknown node, i.e., the topology database includes all *-subpaths $(n_i, *_v, n_j)$ where $i, j \in [1, 3], i \neq j$. Figure 4.4-b presents an example of this case where the data set includes *-subpaths among all known neighbors of the unknown node as shown in Figure 4.4-a. Note that larger cliques may appear in collected path traces however in our earlier study we observed their occurrence frequency to be very low [60]. As searching maximal cliques is NP-hard, in this study, we limit cliques to 3-cliques, i.e., triangles.

We index all triangles in the graph using *Algorithm 9* in Figure 4.10. The algorithm iterates over the star structures reported by *Algorithm 8* in Figure 4.8. Since we only process *-subpaths of length 3 (i.e., 2 known nodes and an unknown node) with triangle structure, we consider only the star structures having $l = 1$. In the algorithm, for each star structure $s_{(a,1,N,M)}$, and $s_{(N_i,1,N^*,M^*)}$ where $N_i \in N$, we obtain the intersection set I of the N and N^* sets. For each $I_i \in I$, $t_{(\{a,N_i,I_i\},1,M)}$ constitutes a triangle where a, N_i, and I_i are the three known nodes of the triangle, and M is the set of all unknown nodes in this triangle structure. Indexing all triangles takes

$O(|S|.a_N.log(a_N))$ where $|S|$ is the number of star structures, and a_N is the average l-neighbor set size of all stars.

4.3.2 Unresponsive Router Resolution

After building graph indexes for star, complete bipartite and triangle structures, we resolve the corresponding unresponsive routers. During the resolution process, we first handle the triangle structures, then complete bipartite structures, and finally the star structures as there is a higher possibility of conflicts within star or complete bipartite structures. That is, it is more likely to have a star structure that is caused by multiple unresponsive routers than a triangle structure.

4.3.2.1 Triangle Resolution

We resolve the triangle structures in the graph using the *Algorithm 10* in Figure 4.11. If trace preservation condition is satisfied, we combine unresponsive nodes A of a triangle structure $t_{(a,b,c,l,A)}$ into a single unresponsive router in the constructed map. Moreover, before the resolution, we sort triangles based on the total number of neighbors and process triangles from the one with the smallest number of neighbors to the largest. This ordering leaves cliquish structures that are due to multiple neighboring unresponsive routers to be processed later. In those scenarios, we might have multiple triangles some of which conflict the trace preservation condition.

4.3.2.2 Complete Bipartite Resolution

Next, we resolve the complete bipartite structures in the graph using the *Algorithm 11* in Figure 4.12. We combine unresponsive nodes A of a complete bipartite structure $k_{(V_1,V_2,l,A)}$ into a single or a chain of nodes in the constructed map under the trace

INPUT: $G = (V, E)$ from Algorithm 6; T from Algorithm 9;
sort (T)
for (each $t_{(\{a,b,c\},l,A)} \in T$)
  if (***mergeable***(A))
    merge$(*_e \in A)$

Figure 4.11: **Algorithm 10**: Resolving Triangle Substructures

INPUT: $G = (V, E)$ from Algorithm 10; K from Algorithm 8;
sort (K)
for (each $k_{(ls,rs,l,A)} \in K$)
  if (***mergeable***(A))
    merge$(*_e \in A)$
  else
    $A^* \leftarrow$ ***findMergable***(A)
    merge$(*_e \in A^*)$

Figure 4.12: **Algorithm 11**: Resolving Complete Bipartite Substructures

INPUT: $G = (V, E)$ from Algorithm 11; S from Algorithm 7;
sort (S)
for (each $s_{(r,l,A)} \in S$)
  if (***mergeable***(A))
    merge$(*_e \in A)$
  else
    $A^* \leftarrow$ ***findMergable***(A)
    merge$(*_e \in A^*)$

Figure 4.13: **Algorithm 12**: Resolving Star Substructures

preservation condition. If the unresponsive nodes in A violate the trace preservation rule, we find the maximal mergeable subset A^* of these unresponsive routers. If there is any, we combine unresponsive nodes in this maximal mergeable subset. Similar to triangle structures, we first sort complete bipartite structures based on their sizes, i.e.

$|V_1| * |V_2|$. We then start processing from the smallest complete bipartite structure to the largest as they have smaller probability of having conflicts.

4.3.2.3 Star Resolution

Finally, we resolve the star structures in the graph using the *Algorithm 12* in Figure 4.13. We combine all unknown nodes A of a star structure $s(a, l, N, A)$, i.e. all unresponsive neighbors of a node a, into a single node in the topology map under the trace preservation condition. If the unresponsive nodes in A do not satisfy the trace preservation rule, we find the maximal mergeable subset A^* of these unresponsive routers. If there is any, we combine the unresponsive nodes in this maximal mergeable subset. In the process, we first sort star structures based on the total number of unresponsive routers they have. We then start processing from the star structure with the smallest number of unresponsive routers to the largest. This way, non-conflicting sets of unresponsive nodes will be processed before the ones that can not be merged into a single node.

4.4 Evaluations

In this section, we use simulations and genuine data to evaluate accuracy and performance of *graph based induction* (GBI) approach to resolve unresponsive routers.

4.4.1 Simulation-based Evaluations

In our simulations, we use a synthetic topology to compare the accuracy of our approach with that of *initial pruning* (IP) [27] and *neighbor matching* (NM) [70] approaches. IP is a commonly used technique and corresponds to the *Algorithm 5* in

Table 4.5: 14% Unresponsive Router Regions for (10,2000) T-S Sample

	Avg. Unresponsive Router Ratio	Avg. Edit Distance
Initial	60.1%	23,122
IP	6.21%	1,726
NM	3.78%	1,403
GBI	1.83%	718

Figure 4.5. NM is similar to the star resolution step in Figure 4.13. Note that we did not compare GBI with graph minimization [128], dimensionality reduction [70] and spectral embedding [15] approaches due to their high complexities (see Section 4.4.3 for details). We also consider the impact of unresponsive router resolution on generated topologies.

For our comparisons, we use *Synthetic transit-stub (T-S) network* generated by GT-ITM topology generator [130]. This network consists of 50,000 nodes and 138,500 links. We utilize it as the underlying actual network. We randomly select a number of nodes in the network as unresponsive, i.e., we choose the unresponsive routers using independent Bernoulli trials, and collect a number of (k,m) path traces to reflect traceroute-collected path traces. In order to asses the effect of unresponsive regions as reported in Table 4.2, we generated synthetic topologies where 14% of routers were unresponsive while they followed the *-subpath distribution of 2011 Ark dataset. Note that in this study, we only consider *permanently unresponsive* routers as IP and NM approaches are completely ineffective for *temporarily unresponsive* routers.

We use *Edit distance* and *unresponsive router ratio* as metric to assess the accuracy of each approach. Table 4.5 presents the results for (10,2000) T-S sample which indicate GBI performs much better than the other approaches.

Table 4.6: Graph Based Induction Technique on Real Data Sets

	#Traces	#IPs	#Unk. nodes	Alg. 5	Alg. 6	Alg. 10	Alg. 11	Alg. 12	Total resolved	Final Unk.
iPlane	32M	300K	15,182,604	14,320,510	212,460	3,212	158,862	406,101	15,101,944	80,660
Ark	22.7M	1.2M	7,862,649	7,213,793	122,820	2,688	115,194	244,076	7,698,571	164,078
Cheleby	15M	1.2M	7,207,885	6,137,750	251,279	2,858	143,880	419,204	6,954,971	252,914

Table 4.7: Impact of Alias Resolution

	No Alias Resolution		Perfect Alias Resolution		APARv2	
	2%	14%	2%	14%	2%	14%
Initial	4,212	17,726	3,426	13,419	3,712	15,127
IP	158	1,597	140	1,293	146	1,346
NM	68	1,372	30	1,096	33	1,213
GBI	30	732	21	645	24	658

4.4.2 Impact of Alias Resolution

Internet topology mapping studies mainly involve IP alias resolution and unresponsive router resolution tasks. Each of these tasks might have inaccuracies, i.e. false positives and/or false negatives. In order to analyze the impact of performing IP alias resolution before unresponsive router resolution, we measure edit distance values with and without alias resolution for (10,2000) T-S sample. For this experiment, we used both an ideal alias resolution where all aliases are resolved and partial resolution with APARv2 [62, 75] that provided 65% to 70% resolution on average. We also varied the unresponsive router ratio in the topology, in order to analyze the changes with different unresponsiveness.

Table 4.7 presents the edit distance values for initial and final topologies. Applying alias resolution before unresponsive router resolution improves the edit distance values for all approaches. Even though the improvement in edit distance is higher for IP and NM approaches than GBI approach, GBI produces the best edit distance results for all cases.

4.4.3 Experimental Results

In this section, we use genuine topology data to analyze the practicality of GBI. We use three data sets collected in February, 2011:

(i) iPlane: 234 vantage points to ∼144K destinations [87];

(ii) Ark: 51 vantage points to ∼9.5M destinations [2]; and

(iii) Cheleby: 400 vantage points to ∼3.5M destinations [75].

After filtering inaccurate and incomplete path traces, we resolve IP aliases using APARv2 [62, 75]. Table 4.6 presents the results for each step of GBI to resolve unresponsive routers in the data set. *Algorithm 5* applies the initial pruning to considerably reduce the number of unknown nodes. *Algorithm 6* then identifies unknown nodes due to temporarily unresponsive routers (i.e., the ones due to ICMP rate limiting or due to congestion at the router). This step resolves over 27% of the existing unknown nodes. Note that none of the previous approaches handle this type of unresponsiveness. We then index star, complete bipartite and triangle structures using *Algorithm 7* , *Algorithm 8*, and *Algorithm 9*, respectively. Next, *Algorithm 10* addresses the unknown nodes in the triangle structures. Note that due to the (k,m) nature of the the traces that span from a few sources to a large number of destinations, the number of observed clique structures and the corresponding resolutions is small. In the following step, *Algorithm 11* resolves the unknown nodes in the complete bipartite structures. Finally, *Algorithm 12* processes the unknown nodes in star structures.

Overall, GBI reduces the number of unknown nodes by 99%, 98% and 97% for iPlane, Ark and Cheleby data sets, respectively (and by 91%, 75% and 76% if we consider the topology after initial pruning as the starting point). Note that no resolution process will reduce the number of unknown nodes by 100% when there

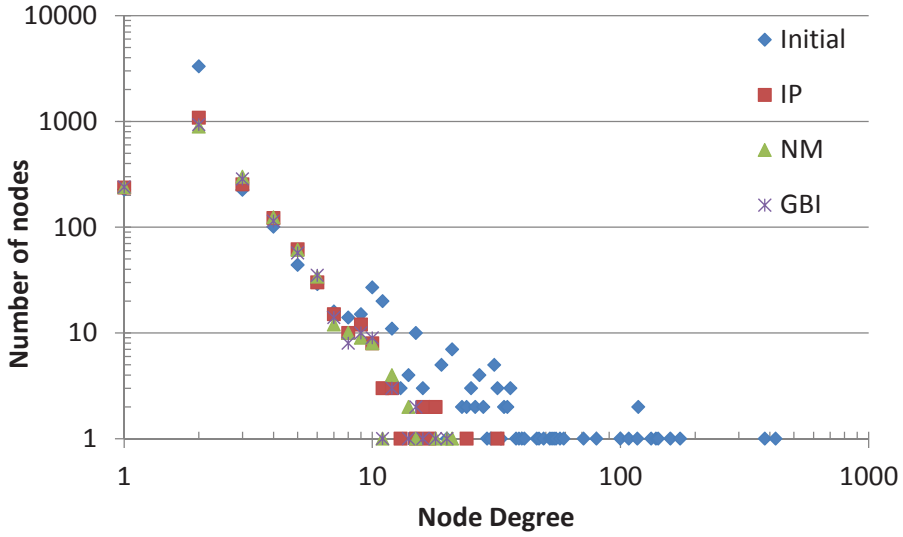


Figure 4.14: Degree distribution

are permanently unresponsive routers. Moreover, we observe that resolution is better with denser graphs such as iPlane that focuses traces on a region. Figure 4.14 presents the degree distribution of the resulting topologies. As shown in the figure initial topology is considerably different from any of the resolved graphs. Moreover, initial pruning inflates node degrees.

Finally, we briefly examine the operational overhead of GBI in Table 4.8. We estimated the number of required operations by using the data structure sizes presented in Table 4.9. We use these values to compare the run time overhead of GBI with earlier approaches. Based on the run time of the GBI algorithms, the highest time complexity is due to the *Algorithm 6*, i.e., approximately $5.2*10^7$ operations for the iPlane data.

The NM approach has a time complexity of $O(n^2)$ where n is the total number of nodes in the data set after the initial pruning. For iPlane data, n is approximately 1.15M and hence NM would take 10^{12} steps. Similarly, the dimensionality reduction approach of [70] would take 10^{18} operations while the graph minimization approach

Table 4.8: Complexity and Operational Overhead of GBI

	Time Complexity	Number Of Operations		
		iPlane	Ark	Cheleby
Algorithm 5	$O(\|U\|.log(\|U\|))$	5.0 * 10^6	2.2 * 10^6	6.7 * 10^6
Algorithm 6	$O(\|U\|.(a_{nd})^2)$	**5.2 *** 10^7	1.4 * 10^6	**9.7 *** 10^6
Algorithm 7	$O(\|V\| + \|U\|)$	0.9 * 10^6	1.2 * 10^6	1.2 * 10^6
Algorithm 8	$O(\|S\|.(a_N)^2)$	1.1 * 10^7	**4.1 *** 10^6	5.6 * 10^6
Algorithm 9	$O(\|S\|.a_N.log(a_N))$	1.3 * 10^6	0.5 * 10^6	0.7 * 10^6

Table 4.9: Size of the Data Structures

	iPlane	Ark	Cheleby
$\|U\|$	0.9M	0.4M	1.1M
$\|V\|$	0.3M	1.2M	1.2M
$\|S\|$	219K	93K	148K
$\|K\|$	858K	752K	793K
$\|T\|$	99K	4.7K	5.3K
a_{nd}	7.67	3.55	3.11
a_N	7.07	6.62	6.11

of [128] would take 10^{30} operations. Since Almog et al. [15] haven't provided the time complexity of their approach, we are not able to compare the computational overhead of their approach. However, they utilize a distance matrix to resolve the unknown nodes and the straightforward resolution of a large scale Internet map is not practical with this approach. Authors partition the map into subgraphs and handle each separately.

4.5 Summary

In this chapter, in order to asses the extend of unresponsive routers in Internet topology mapping studies, we first present an experimental study on the responsiveness of routers to active probe messages. In our historical analysis, we observe that responsiveness reduced during the last decade and regions of unresponsive routers exists.

We also observe that network operators are increasingly using rate limiting of active probes. Moreover, we observe that routers are less willing to respond to direct probes as compared to indirect probes and the responsiveness of routers changes with the type of the probes: ICMP based probes having the highest response rate and UDP based ones having the lowest. Even though TCP based probes receive responses much better than UDP based ones, this type of probes are more likely to raise security alerts.

In the second part of the chapter, we develop a *Graph Based Induction* approach to resolve unresponsive routers. In this approach, based on our novel structural graph indexing, we index observed subgraphs that contain unknown nodes. Then, we determine the corresponding minimal underlying structure that satisfies the trace accuracy condition. Our work improves the state of the art in unresponsive router resolution in terms of both accuracy and efficiency. Regarding accuracy, GBI addresses all cases of unresponsive routers whereas the previous approaches ignore temporary unresponsiveness. Regarding efficiency, the run time complexity of our algorithm is significantly less than that of existing algorithms. Our experiments on three different data sets have shown a significant reduction in the practical run time overhead of our approach (approximately, $5.2*10^7$ operations) as compared to the previous approaches (approximately, 10^{12}, 10^{18}, or 10^{30} operations), in the worst case.

Chapter 5

Cheleby: An Internet Topology Mapping System

In this chapter[1], we present Cheleby, an Internet topology mapping system that provides insight into the Internet topology by taking daily snapshots of the underlying networks. The system utilizes efficient algorithms to process large scale data-sets collected from distributed vantage points and provides accurate topology graphs at link layer. Incorporating enhanced resolution algorithms, Cheleby provides comprehensive topology maps.

[1]Preliminary versions of this chapter appeared in:

Hakan Kardes, Talha Oz, and Mehmet H. Gunes *Cheleby: A Subnet-level Internet Topology Mapping System*, 4th International Conference on COMmunication Systems and NETworkS (COMSNETS), Banglore, India, Jan 3-7, 2012.

Hakan Kardes, and M.H. Gunes. *Subnet-level Internet Topology Mapping: Issues and Resolution Methodologies.* Elsevier Computer Networks. (under review)

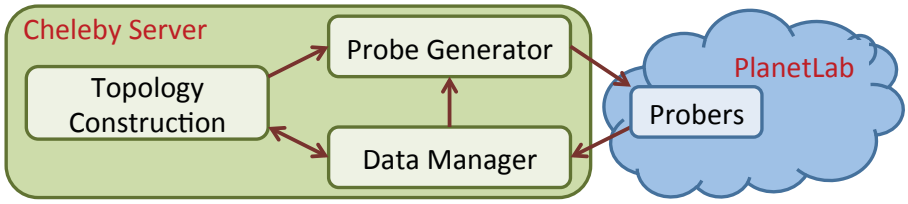


Figure 5.1: Cheleby System Overview

5.1 Introduction

Cheleby topology mapping system, shown in Figure 5.1, runs on a server which actively manages PlanetLab nodes [8] as its monitors to collect topology information from geographically diverse vantage points. The server instructs monitors to collect partial path traces and perform other probing activities. Cheleby then resolves subnets, IP aliases and unresponsive routers within the collected raw data to construct the network graph corresponding to the sampled network.

In Section 5.2, we present mechanisms to gather information from the Internet and to obtain accurate samples with minimal traffic overhead. We also present an overview of the Cheleby topology sampling system and discuss experimental results with various system parameters. In Section 5.3, we present topology construction methodology we used in Cheleby. Finally, in Section 5.4, we conclude the chapter.

5.2 Topology Sampling [2]

In order to sample the underlying topology of the Internet, we collect a large number of path traces from geographically diverse vantage points towards all /24 subnets in the Regional Internet Registries (RIR). An important issue affecting accuracy of collected path samples is that certain traffic engineering practices for load balancing

[2]This section is a joint work with Talha Oz.

may cause traceroute to return IP addresses that do not correspond to a real end-to-end path in the Internet. We utilize Paris traceroute, which fixes flow identifiers so that flow-identifier based load balancing routers will choose the same next hop for probe packets toward the same destination [20]. Moreover, we perform ICMP based querying as it elicits more responses than other probing approaches [61]. In the following, we describe major steps of Cheleby regarding topology sampling.

5.2.1 Destination List Generation

In order to probe active subnetwork ranges, we obtain subnet announcements with originating AS from `http://www.cidr-report.org`. The list provides updated advertisements and actual RIR allocations for each AS[3]. For each subnet range, we pick the first allocatable IP address in the range. If the subnet is larger than /24, then we divide the range into /24 subnets (e.g., $A.B.C.0/24$) and pick the first allocatable IP address as the probing destination (i.e., $A.B.C.1$). These IP addresses are then divided into blocks of approximately 1,024 destinations that will be probed by monitors. At the end of this process, we have 3,460 destination blocks, i.e., **3.54M** destination IP addresses to start with.

Moreover, as probing cycles are completed, we replace non-observed IP addresses with responsive IP addresses, which have a common subnetwork prefix of /24 or longer, in the earlier data sets. Additionally, we dynamically append newly observed router IP addresses to the destination lists during the topology construction phase (see Section 5.3.5).

[3]Note that we may integrate additional data sources such as Route Views

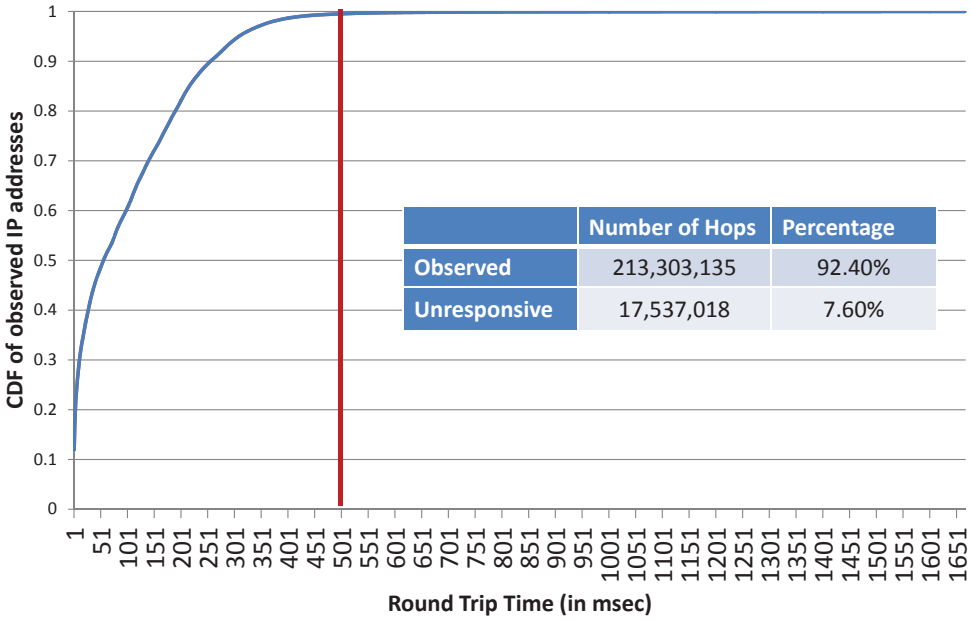


Figure 5.2: Cumulative Distribution Function for RTT

5.2.2 Response Wait Time

In order to determine time-out time for traceroute probes, we analyzed the response time of elicited responses for traces towards 3.54M destinations using a time-out of 1.7 seconds. Figure 5.2 presents the cumulative distribution function of the round trip time (RTT) for probes that elicited an ICMP response. In this experiment, we elicited a response to 213.3M probes and had 17.5M probes without a response. In Figure 5.2, we observe that more than 99.95% of responsive nodes had an RTT less than 0.5 sec. Hence, in subsequent experiments we set the time-out to 0.5 sec since longer time-outs delay the overall topology collection process.

5.2.3 Task Assignment to Monitors

In order to probe destinations from geographically diverse vantage points, Cheleby utilizes the PlanetLab nodes around the world. Among ∼1,100 nodes only ∼600 of

them were functional during our experiments. As the Paris traceroute did not run on ∼100 of functional monitors, we could utilize ∼500 nodes during our topology collection.

Figure 5.3 displays clusters of monitors for 5 and 7 teams. For instance, for 7 teams, shown with green squares, functional PlanetLab nodes are divided into teams as 1: North-West America, 2: North-Central America, 3: North-East America, 4: South America, 5: Western Europe, 6: Eastern Europe + Africa + Western Asia, and 7: Eastern Asia + Australia. Similarly, the distribution for 5 teams is indicated with blue dashed lines. Ark utilizes a similar approach to divide its 53 monitors into three teams [2]. The main difference between Cheleby's and Ark's task assignment strategy is that Cheleby utilizes available monitors in dynamic fashion (as described below) while Ark utilizes dedicated systems.

Cheleby dynamically assigns one of the available monitors from each team to probe destination blocks. Each destination block is probed by only one monitor at a time and overall as many times as the number of teams. Each monitor is set to probe 4 destination blocks in parallel to reduce the overall round completion time. Each of the 4 monitor processes work independent of others. These processes are marked as *idle*, *busy*, or *inactive*. All processes in a monitor is *inactivated* when one of them returns its data in less than 2 minutes as this indicates a problem with the probing. Monitors are ranked based on their average task completion times and Cheleby selects the top *idle* process from a team to assign a new destination block. Monitors can be ranked based on other metrics such as the number of unique edges and/or nodes discovered in previous task but a large/responsive AS probed earlier may cause biases.

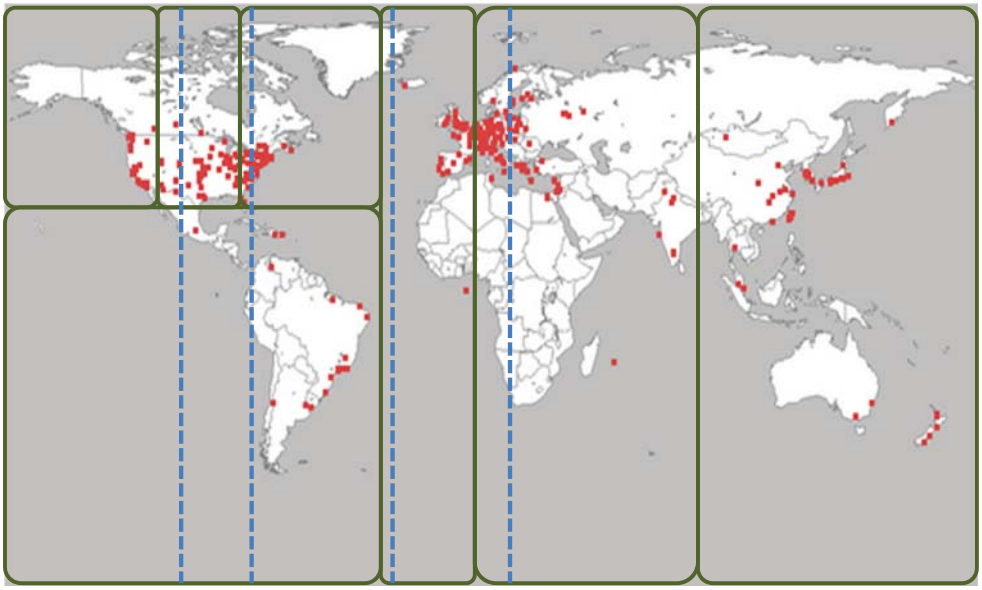


Figure 5.3: Team assignment of PlanetLab nodes (Blue lines: 5 teams. Green boxes: 7 teams.)

Probing of a monitor is terminated if the monitor cannot complete its task within a period of 2 hours. In this case, the monitor's ranking is reduced and brought to the *idle* state. The partially traced destination block is also put to non-probed list for another trial by a second monitor in the same team. If the new monitor, which reverses the order of destination IP addresses before probing, is not able to complete probing in time as well, then the destination block is marked as partially completed and both of the partial traces are appended to the database.

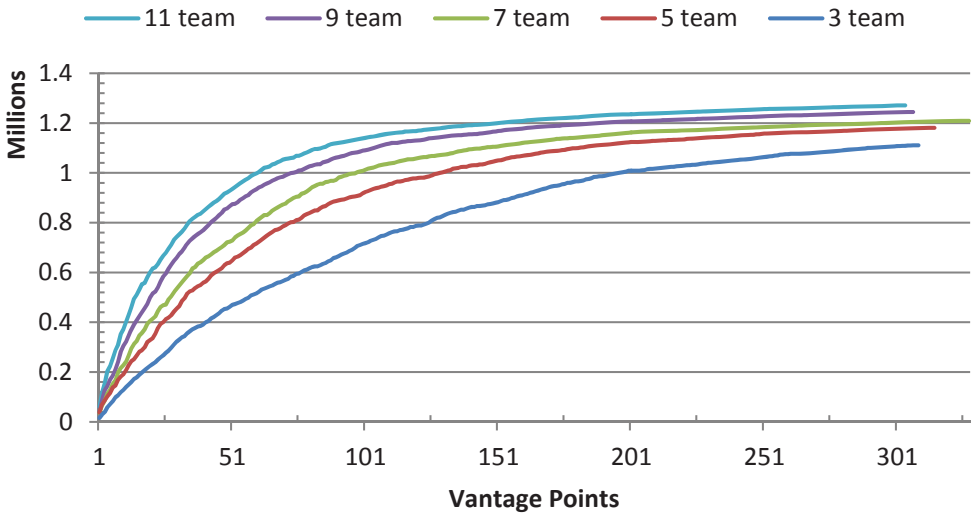
We performed an experiment to analyze the effect of choosing different number of teams by varying team sizes. Variations in the number of teams has a direct effect as seen in Table 5.1, which, for different team configurations, presents: (1) the round completion time, (2) generated traces, (3) generated probes, (4) probes yielding an IP address, (5) probes that did not elicit a response, (6) unique IP addresses, (7) percentage of observed IP addresses compared to combination of IP addresses from

Table 5.1: Team Statistics with Different Team Sizes

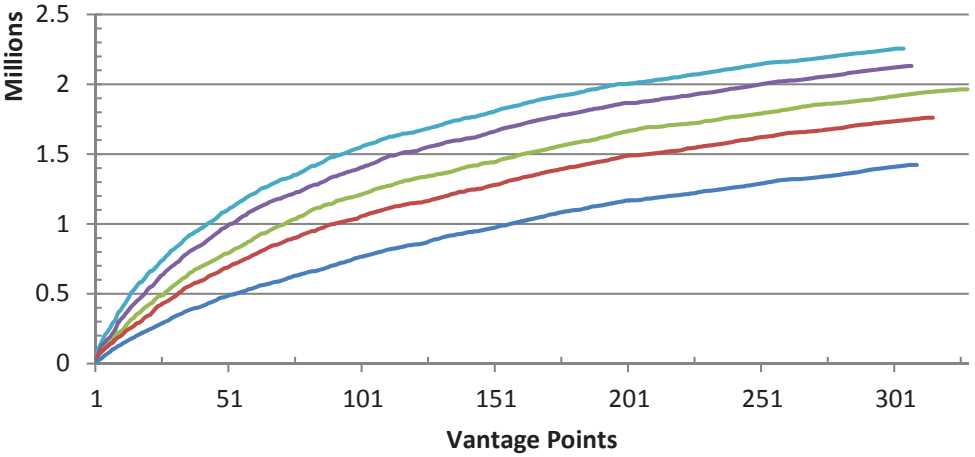
Teams	3	5	7	9	11
Time (min)	540	630	770	1,220	1,540
Traces	9.5M	15.9M	22.0M	28.7M	35.0M
Probes	151M	249M	347M	452M	552M
Total IPs	95.3M	157M	219M	285M	348M
Total *s	55.7M	92.4M	128M	167M	204M
Unique IPs	1.11M	1.18M	1.21M	1.23M	1.24M
IPs / all	79.3%	84.3%	86.3%	88.8%	90.7%
Per min IPs	2,057	1,874	1,571	1,020	825
Unique Edges	1.42M	1.76M	1.96M	2.06M	2.10M
Edges / all	46.1%	57.1%	63.6%	69.1%	73.1%
Per min Edges	2,636	2,794	2,550	1,747	1,465

all experiments in the table, (8) observed IP addresses per minute of probing, (9) unique edges, (10) percentage of observed edges compared to union of edges from all data sets, and (11) observed edges per minute of probing. Additionally, Figure 5.4 presents the changes in the number of observed IP addresses and edges with aggregate monitor data.

As the number of teams increases, more probes are generated and fewer monitors are deployed per team. Both of these cause longer round completion times. However, as seen in the unique IPs and unique edges rows, there is a diminishing benefit with higher number of probes (as reported by [] as well). Even though, using 11 teams returns the highest number of IPs and edges, the overhead is highest per observed IP address. An important observation is that the overlap between the edges is much smaller than the overlap between the IP addresses because the deployed monitors in each case differ. Considering this analysis, we utilized 7 teams in the remaining experiments as it provides the best balance between coverage and overhead.



a) Number of Known Nodes



b) Number of Edges

Figure 5.4: Number of Nodes and Edges for different team sizes

Using 7 teams, we performed 8 rounds of data collection to observe team dynamics. Table 5.2 presents the averages of (1) monitors, (2) incomplete destination blocks, (3) completed destination blocks, (4) destination blocks that could complete in the second trial, (5) average block completion times in seconds, and (6) total run time for each team in hours. Initially, we clustered the monitors around the world into regions to balance the number of monitors per team. However, teams 5, 6, and

Table 5.2: Team Statistics (Average of 8 Data Sets)

Team	#1	#2	#3	#4	#5	#6	#7
Monitors	56.63	53.88	55.50	56.75	77.25	73.63	76.25
Incomplete Dest Blocks	7.43	30.28	24.03	35.72	12.85	12.35	12.15
Completed Dest Blocks	3,453	3,430	3,436	3,424	3,447	3,448	3,448
Completed in 2nd Trial	16.2	63.1	40.6	60.4	26.9	23.4	26.1
Avg. Compl. Time (sec)	1,476	1,376	1,586	1,650	1,764	1,764	1,566
Run Time (hours)	8.53	8.18	9.15	9.32	7.32	7.68	6.54

7 were considerably behind others and we increased their monitors by adjusting the geographic clusters.

As seen in the Table 5.2, on average 19.26 of the 3,460 destination blocks were not completed in the designated time of 2 hours even after the 2^{nd} trial. Team 4 (South America) had the lowest probe completion with an average of 35.72 incomplete destination blocks (i.e., 1.03% of all blocks). On average, 36.67 of blocks were completed in the 2^{nd} trial, which is included in the overall completion numbers. Team 5 (Western Europe) and Team 6 (Eastern Europe + Africa + Western Asia) were the slowest with an average of 1,764 seconds to trace a destination block. However, Teams 5, 6, and 7 were the fastest ones in probing all destination blocks due to the higher number of monitors in these teams. This is also reflected in Figure 5.5, which shows the destination block probe completion times of each team for a single run. Destination blocks in the Figure 5.5 are ranked by the average completion times of all teams for the block from the largest to the lowest (shown with black line). In the figure, we observe that there is a group of destination blocks that complete probing approximately in 700 seconds independent of team averages. This often happens when the destination block is in the same region of the probing team.

Figure 5.6 displays completion statistics for a single data set where monitors for each team are ranked by the number of destination blocks they completed probing.

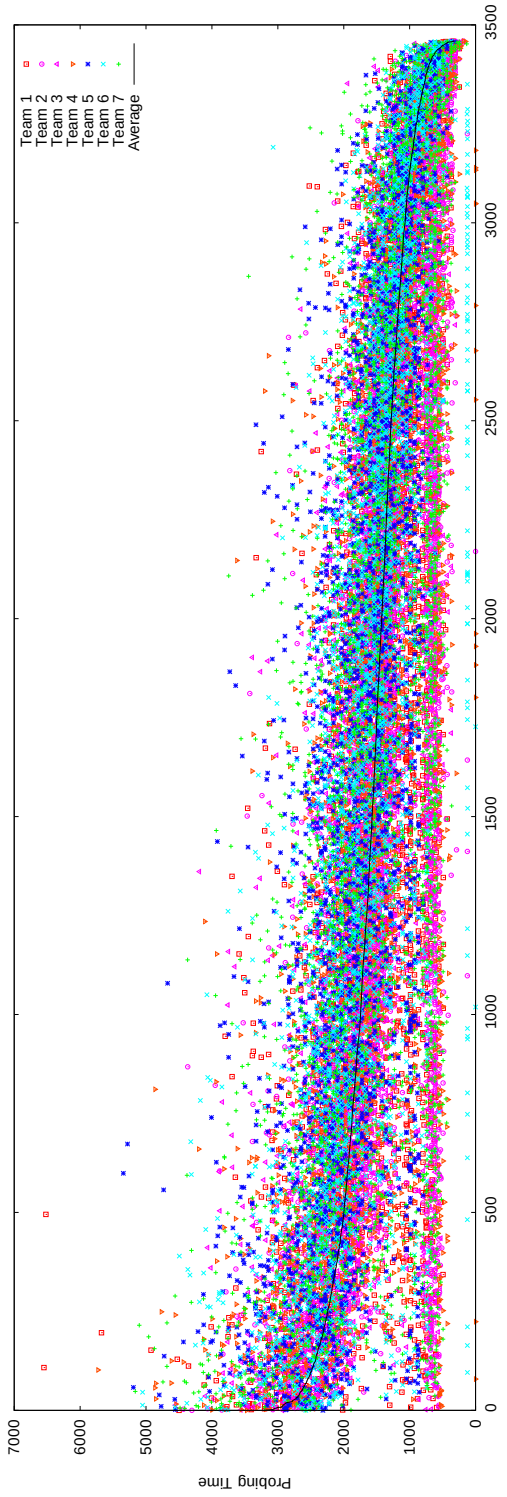
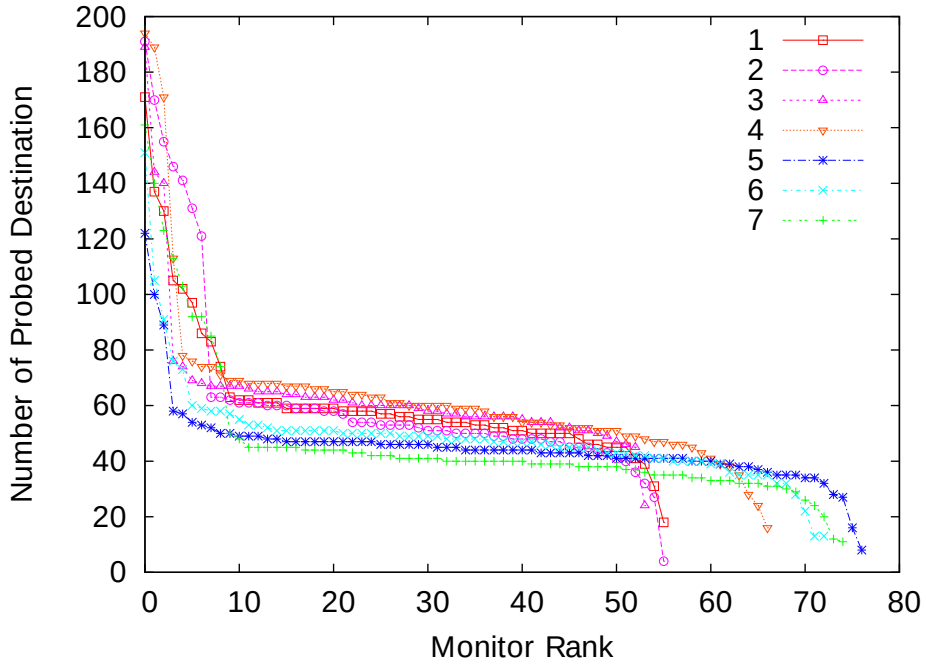
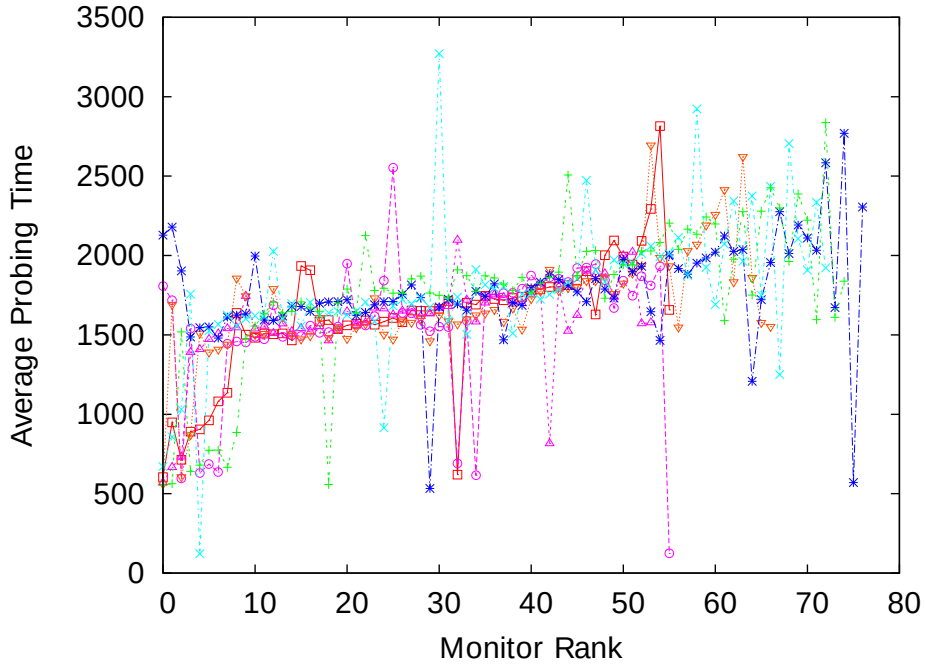


Figure 5.5: Completion Time per Destination Block (in Seconds)



a) Average Number of Probes



b) Average Completion Time (sec)

Figure 5.6: Team Completion Statistics

As seen in Figure 5.6-a, while most of the monitors completed 40 to 80 destinations blocks, there were outliers that either outperformed or fall behind others. Moreover, as seen in Figure 5.6-b, average probe completion times increased in general with lower ranked monitors. In general, the outliers that were considerably below the average curve were faulty monitors that either returned responses in few minutes, whose data was removed and set *inactive* for certain time, (e.g., Team 2 node at 55) or became available for part of the data collection (e.g., Team 5 node at 75). On the other hand, outliers well above the average line received a highly unresponsive destination block, i.e., AS regions that were not very responsive to probes, causing jumps in the task completion time. Overall, the dynamic task assignment helped improve round completion time to less than half of the initial experiments where tasks were randomly assigned without timeouts and penalties.

In Figures 5.7, 5.8, and 5.9, we analyze the number of newly added known nodes, unresponsive nodes, and edges in order to show the effect of the number of vantage points on the completeness of the resulted graph. As the number of vantage points increases, there is always an increase in the number of newly observed unresponsive routers. However, we observed just a few new known nodes after the hundred vantage points. There is an increase in the number of newly observed edges while the number of vantage points increases. Considerable amount of newly added edges are the contribution of the newly observed unresponsive routers.

Finally, Figure 5.10 presents the average of the number of unique nodes and edges observed as data from vantage points and destination blocks are appended to the graph, respectively. Similar to earlier findings, we observe that addition of more monitors sub-linearly increases the number of unique IP addresses or edges. On the other hand, number of unknown nodes increases linearly as the unresponsive routers

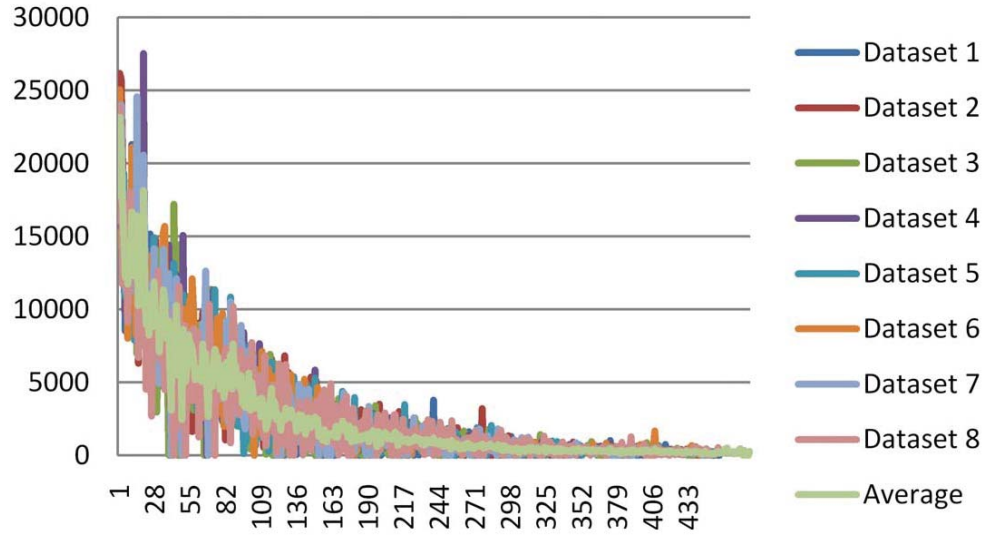


Figure 5.7: The Number of New Known Nodes After Each Vantage Point

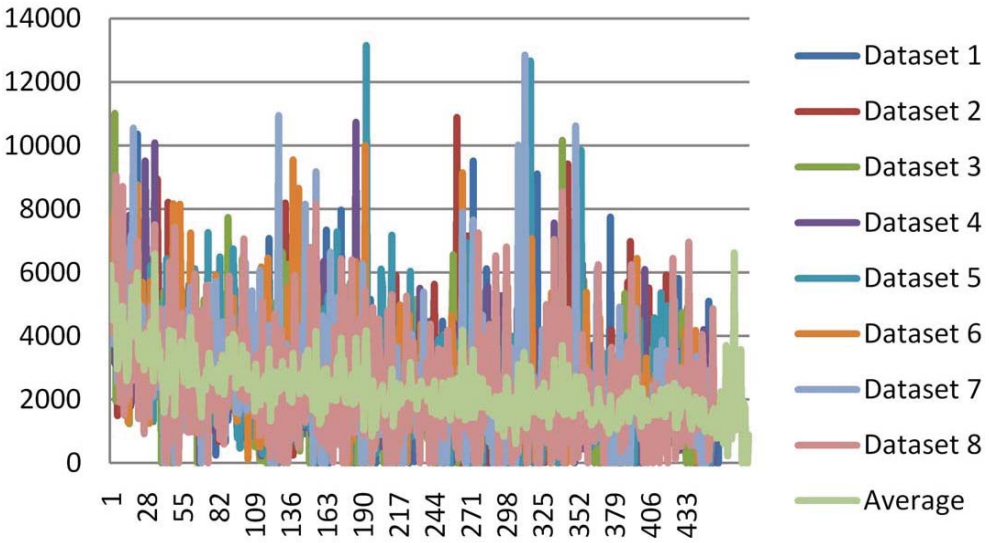


Figure 5.8: The Number of New Unresponsive Nodes After Each Vantage Point

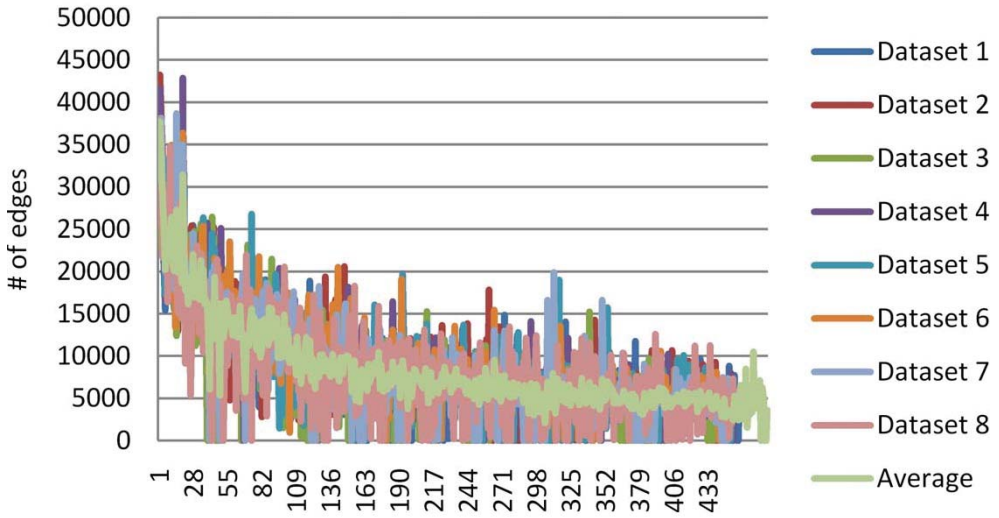
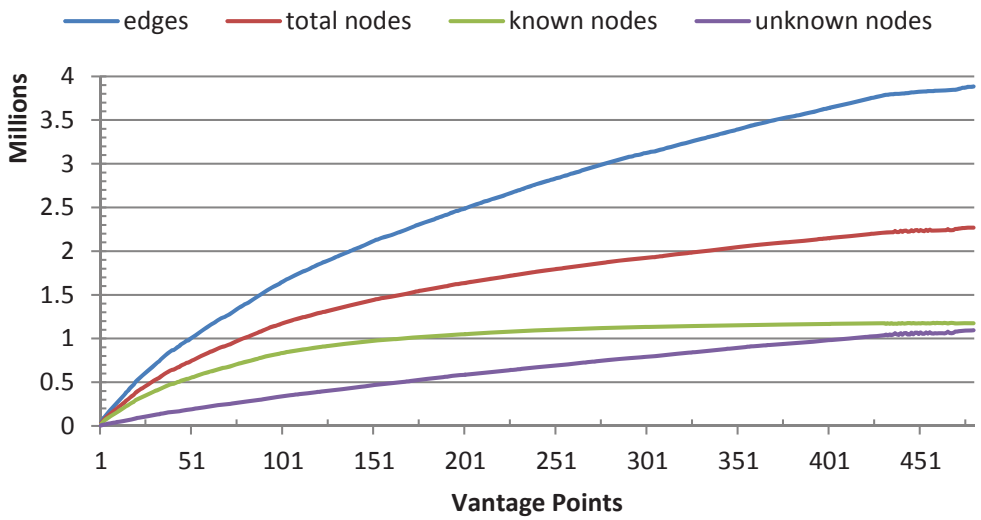


Figure 5.9: The Number of New Edges After Each Vantage Point

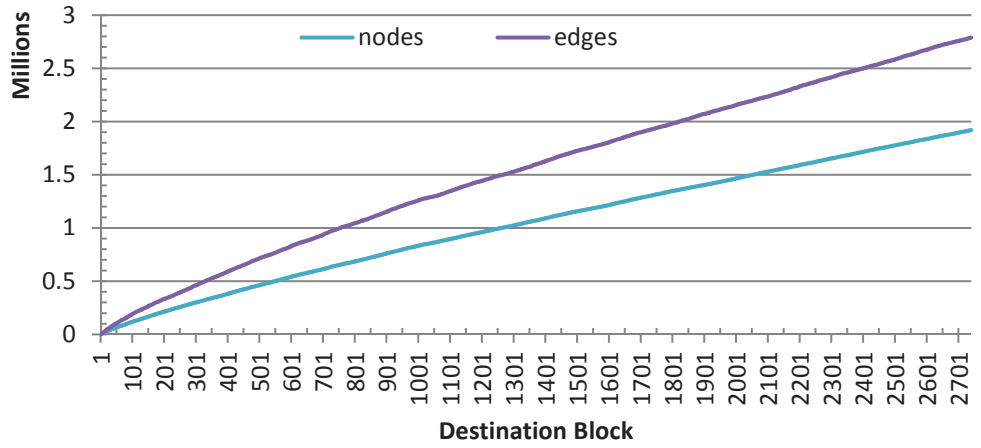
are not resolved yet and each instance is recorded as a unique node. Finally, addition of destination blocks almost linearly increases the number of observed IPs and edges because the destination blocks are towards different ASes.

5.2.4 Probing Overhead Reduction

In Cheleby, we utilize inter-monitor and intra-monitor probe reduction as shown in Figure 5.11. We reduce intra-monitor redundancy by gathering partial traces to some destination IP addresses. Once we have a full trace to an IP address in an AS, we start successive traceroute queries from the hop distance h_i of the ingress router (i.e., hop distance of the last IP address in the trace that doesn't belong to the destination AS). If the first IP of the new trace has not appeared at the same hop distance h_j in any of the earlier full traces to the AS, then we complete the trace. Otherwise, we utilize the partial trace as the remaining part will most likely overlap with an existing trace. Analyzing collected traces, we observe that 35.4% of 22.4M traces are partial traces. This overall saved 66.2M probes that would be generated with full tracing.



a) Cumulative Monitor Nodes and Edges

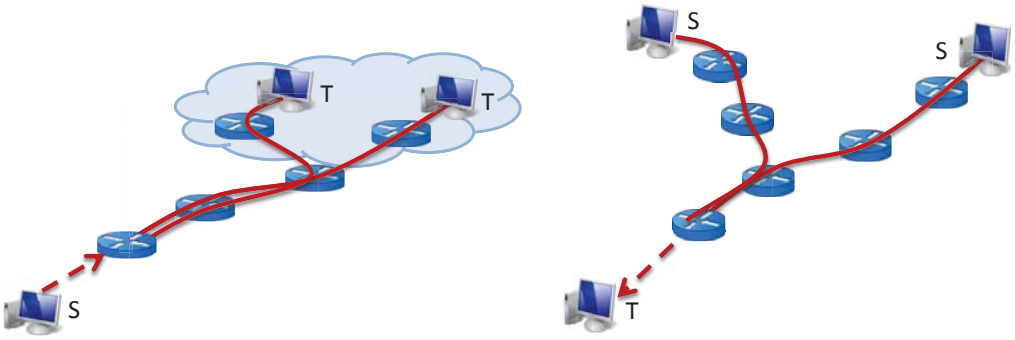


b) Cumulative Destination Block Known nodes and Edges

Figure 5.10: Number of Nodes and Edges (average of 8 data sets)

Note that, this number can be improved because we randomly assign IPs to different destination blocks if an AS has more than 1024 IPs to be traced.

Additionally, to reduce inter-monitor redundant probing, a destination IP is probed by only one monitor of a team. Since the monitors in the same team are geographically close to each other, we expect their contribution to identify a new link/node to be small. Moreover, with additional probing we may identify ingress



a) Intra-monitor b) Inter-monitor

Figure 5.11: Intra- and Inter-monitor Redundancy Reduction

points of ASes to dynamically establish teams for each destination AS so that we have exactly one monitor probing through each ingress point of an AS. That is, we can determine the sets of monitors that probe each ingress point of the AS and then build individual teams for each AS.

Our probe reduction mechanism is similar to Doubletree [42], which models Internet paths as trees. Doubletree uses a local stop set to identify branch points for traces with respect to a source. Our mechanism is similar but instead of starting at a given or random hop h, we make an informed guess of h_i based on the earlier traces. Doubletree additionally uses a global stop set for traces towards a destination. We do not implement this but divide monitors into geographic teams and probe a destination block from one monitor of each team, which effectively reduces redundancies due to tracing of the same path towards a destination.

5.3 Topology Construction

After collecting topology data, we need to process this raw data to obtain the underlying network topology. In Internet topology construction studies, there are several

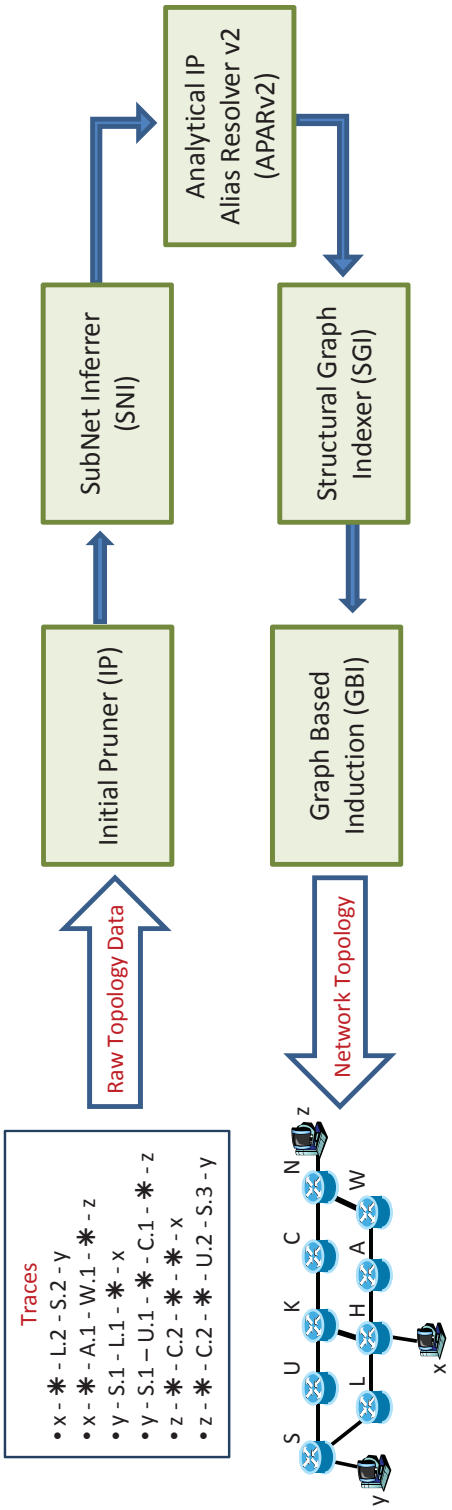


Figure 5.12: Topology Construction

challenges, namely, unknown router resolution, identification of IP aliases and underlying subnets. As we mentioned in Chapter 2,there are several research groups working on these problems. Despite several approaches proposing solutions for one of the these issues, no study considers all these challenges together and proposes an approach to overcome these issues altogether. In Cheleby, first we analyze all of these issues and the relation between them. After our initial experiments on iPlane topology datasets, we found out that in order to resolve unresponsive routers within the Internet topology we first need to identify the IP alias pairs in the topology. Moreover, to reduce the computational complexity, we should infer the underlying subnets before identifying the alias pairs. Therefore, we (1) filter faulty traces, (2) infer underlying physical subnets among IP addresses, (3) resolve IP addresses belonging to the same router, and (4) resolve unresponsive routers as shown in Figure 5.12.

The accuracy and the completeness of these tasks may significantly affect the accuracy of the resulting topology maps [58, 118]. Moreover, when handling these tasks, one needs to make decisions based on the observations to infer the underlying topology. As the earlier decisions effect the later ones, obtaining the most likely topology under various conditions has been shown to be NP-hard [14]. Finally, these resolution tasks especially are challenging when large scale topologies of millions of nodes are processed. In this section, we analyze each of these tasks and present the algorithms that we utilized to handle them efficiently even in very large topologies consisting around 2.5M nodes and about 5M edges.

5.3.1 Initial Pruning

In this step we filter faulty traces, resolve the unresponsive routers in between the same known routers, and constitute our data structures from the raw data.

As path traces contain anomalies such as routing loops, we first prune raw path traces. The pruning breaks path traces with a loop (e.g., IP_A, IP_B, IP_C, IP_D, IP_E, IP_C, IP_F, IP_G) into three pieces based on the repeated IP address (i.e., IP_C) and utilize the first part (i.e., IP_A, IP_B, IP_C) and the last part (i.e., IP_C, IP_F, IP_G) of the trace in the remainder of processing. In collected path traces, 772K (%3.45) of path traces contain routing loops among which 143K has multiple loops. Moreover, we observed border firewalls that filter ICMP packets from/to a network domain and occasionally respond with their IP address. However, the hop distance of these IP addresses are not consistent. Hence, we filter any IP address that appears at the end of trace after 3 unresponsive nodes.

Then, we build initial network graph by parsing filtered path traces. During parsing, we resolve unknown nodes that are between the same set of known nodes. In order to resolve this type of unresponsive router, we need to detect the same *-substrings (i.e., same length *-substrings with the same known nodes at the end points). While reading the traces from the raw data, we identify the unresponsive routers between two known nodes. We read all *-substrings from the database and construct a new graph $\bar{G} = (\bar{V}, \bar{E})$. We represent each *-substring as an edge e(a,b,l) where a is the first known node, b is the second known node and l is the label of the edge representing the number of unresponsive nodes between a and b as in Figure 5.13. We add each $e(a, b, l)$ only once to our new graph $\bar{G}$. In the successive resolution tasks, we use the graph $\bar{G}$. Performing this unresponsive router resolution step during graph construction reduces the number of unknown nodes by %78.71 on average. Additionally, we add all observed unique known and unresponsive routers to $\bar{V}$ and we constitute a neighbor set for each router.

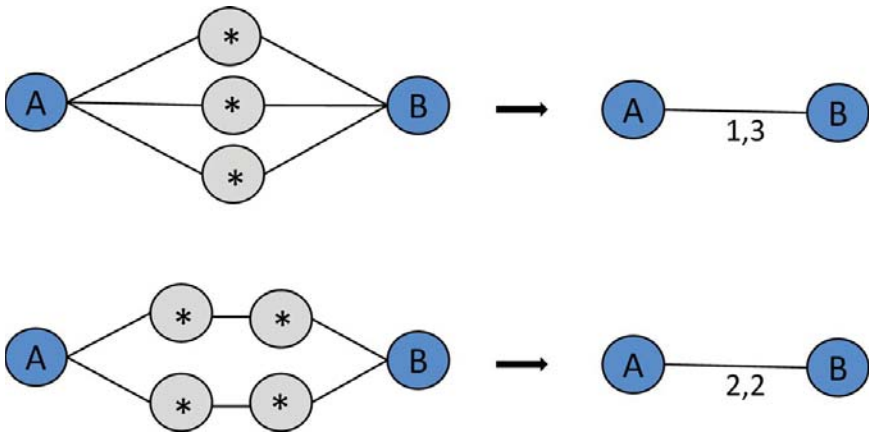


Figure 5.13: Sample Transformation

Table 5.3: Topology Data (in millions)

Data Set	1	2	3	4	5	6	7	8
All Traces	22.39	22.42	22.42	22.40	22.42	22.41	22.42	22.03
Partial Traces	8.02	8.12	8.05	7.86	7.67	7.98	7.91	7.80
Saved Probes	65.23	66.14	67.68	66.32	63.98	67.90	66.19	65.98
Unknown Nodes	4.93	4.81	4.90	4.88	4.95	4.94	4.95	4.92
Known Nodes	1.18	1.18	1.19	1.17	1.20	1.17	1.19	1.17

This process can be called initial pruning (IP). After this step, we sequentially infer subnets, resolve IP alias pairs, and identify the unresponsive routers by first indexing star, complete bipartite, and triangle structures with the SGI algorithm presented in Section 3.1, and then resolve unresponsive routers within these structures with our graph based induction algorithms.

Table 5.3 presents the statistics of (1) all traces, (2) partial traces, (3) saved probes by using partial traces, (4) Unknown nodes, i.e., '*', and (5) Known nodes, i.e., IP addresses, for the analyzed data sets.

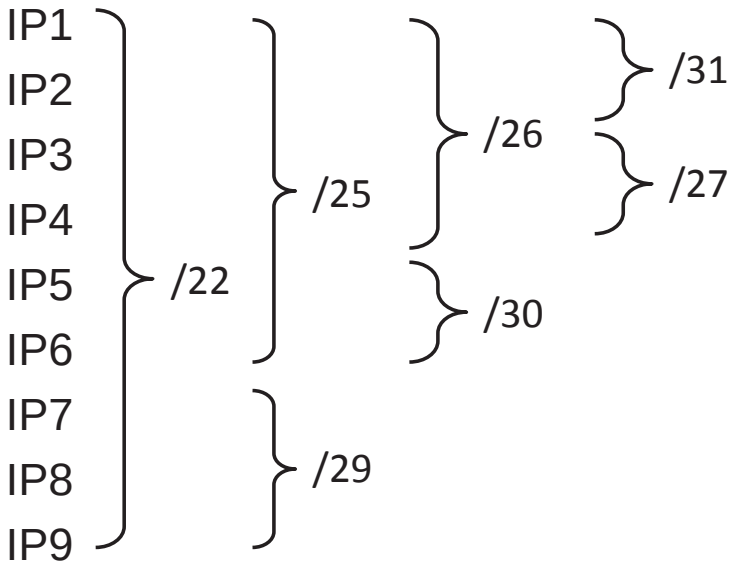


Figure 5.14: Subnet Resolution

5.3.2 Subnet Inference

The first task after building an initial network graph is the identification of the underlying physical subnets, i.e., link level connectivity, among IP addresses in the collected topology [59]. The goal in subnet resolution is to identify multiple links that appear to be separate and combine them to represent their corresponding single hop connection medium (i.e., multi-access link). Subnet resolution also finds missing links between IP addresses that fall in the same subnet range but were not observed in path traces. The successful inclusion of subnet relations among the routers yields topology maps that are closer, at the link layer, to the sampled segments of the Internet.

Cheleby, enhances the subnet resolution approach presented in [59] by utilizing only the *distance preservation* condition but not the *trace preservation condition* to reduce the computational complexity. SubNet Inferrer module (SNI) observes distances of all IP addresses per vantage point and determines IP address ranges that have similar distances to all vantage points. First, it clusters IP addresses into

Table 5.4: Average Subnet Statistics for 8 Data Sets

Subnet Size	/24	/25	/26	/27	/28	/29	/30	/31
Count	0.38	4.25	34.13	485	6,381	20,602	11,202	2,960
Completeness	27.7%	24.5%	23.3%	23.3%	24.8%	36.0%	100%	100%
All IPs	26	131	492	3,383	22,110	44,500	22,403	5,920

candidate subnets up to a given maximal size (e.g. /22). Then, it break them down as necessary as in Figure 5.14 by doing distance analysis on candidate subnets. Similar to [121], we only allow one IP address being closer to each of the vantage points. As the number of vantage points is increased, the distance condition can more accurately filter false subnets without relying on the trace accuracy condition. Furthermore, TraceNET tool enhances traceroute by identifying subnets between a source and a given destination but it focuses on single path traces [119].

Table 5.4 presents statistics of identified subnets and the completeness values of the subnets that had 20% of their IP addresses present in the data set. This number is less than expected as only 99K of collected 1.2M IP addresses appear in a subnet. The main reason for this is because we did not explore other IP addresses of candidate subnets. Thus, we included a probing function into the SNI that probes subnets that have less than half of their IP addresses present in the data set (explained in Section 5.3.5).

5.3.3 IP Alias Resolution

After inferring underlying subnets, Cheleby resolves IP aliases. Since routers have multiple interfaces with different IP addresses, different path traces may include routers with different IP addresses. Hence, we need to identify and group IP addresses belonging to the same router. Without IP alias resolution, the resulting topology map may be significantly different from the actual topology [58].

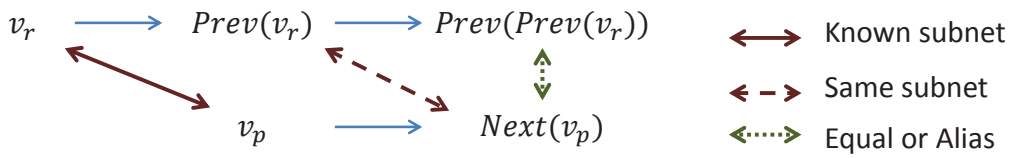


Figure 5.15: Analytical and Probe-based Alias Resolver v2 (APARv2)

Gunes et al. presented the *Analytic and Probe-based Alias Resolver (APAR)* in [62]. Given a set of path traces, the analytical component utilizes the common IP address assignment scheme (see RFC 2050) to infer IP aliases. It uses inferred subnets to align symmetric segments of different path traces and identifies alias pairs among involved IP addresses. Path asymmetry is a commonly observed characteristic in the Internet. However, APAR does not require complete path symmetry and relies on symmetric path *segments* to resolve aliases.

We developed APARv2,an enhanced version of APAR by eliminating path queries as shown in Figure 5.15,i.e., the most significant improvement of APARv2 over APAR is reduction in required storage of path traces. In APARv2, we process all subnet IP address pairs v_p and v_r and determine candidate alias pairs. Then, we verify whether our candidate alias pair (i.e., v_p and $Prev(v_r)$) has a common neighbor (i.e., $Prev(Prev(v_r))$ and $Next(v_p)$) as an alias or as in another subnet relation (i.e., $Prev(v_r)$ and $Next(v_p)$). If common neighbor condition is satisfied, we analyze whether our candidate alias pair appeared in the same trace or not. In order to ensure the accuracy condition without storing all path traces, we store the conflict sets, i.e., set of traces an IP address appeared in, for each known node. These changes help us eliminate the need to keep complete path traces in memory for alias resolution.

Table 5.5 presents average alias resolution statistics for collected datasets. Utilizing APARv2 on collected data, we identified 23,266 alias sets that include 75,019

Table 5.5: Alias Resolution Averages

Alias Sets	IPs in Alias Sets
23,266	75,019

aliased IP addresses on average. However, this corresponds to only ∼7% of observed IP addresses. This value was especially low as we did not include IP-mates (i.e., /30 or /31 pair of the observed IP address) and we had a low subnet coverage as these subnets help in alias identification. Hence, we (1) improve subnet coverage with probing candidate subnet IP addresses, (2) integrate IP-mate probing component into APARv2, and (3) implemented probing based mercator and ally approaches to complement APARv2 as described in Section 5.3.5.

5.3.4 Unresponsive Router Resolution

Unresponsive routers are routers that are passive to measurement probes and are represented by a '*' in a traceroute output. Since a router may appear as a '*' in multiple traceroute outputs, we need to identify '*'s (i.e., unresponsive nodes) that belong to the same router. Even a small number of unresponsive routers may significantly distort the constructed topology [60]. Moreover, the mere volume of unresponsive nodes in the collected data set introduces additional challenges in building an efficient solution.

In Cheleby, we utilize a *Graph Based Induction* (GBI) technique that we detailed in the previous section to resolve unresponsive routers. Graph based induction is a technique to obtain information from a graph in data mining field [90]. We first analyze the nature of unresponsive routers and identify different types of unresponsiveness. Examining a number of topology maps that are constructed from traceroute data with unresponsive routers, we identify a number of graph structures (i.e., paral-

lel, star, complete-bipartite, and triangle) that are formed among unresponsive nodes and their known neighbors. We then develop efficient algorithms to detect these structures in the graph and reduce the unresponsive nodes (i.e., the occurrences of '*'s) into their corresponding unresponsive routers.

Moreover, we enhance the graph based induction technique with our *structural graph indexing* (SGI) [74] approach. Graph indexing is a technique to enhance graph processing time in very large graphs when there are many queries to look for in the underlying topology. Similarly, the structural graph indexing indexes a given set of substructures, i.e., star, complete-bipartite, and triangle in our case, in the sampled graph so that subsequent queries become faster. Graph indexing significantly reduces processing time of the graph based induction to resolve unresponsive routers in the Cheleby system. SGI indexes maximal graphs that match the structure formulation within the original graph in a consecutive manner. SGI first identifies star structures, then complete-bipartite, triangle and finally clique structures from the preceding ones. In our experiments, we realized that the number of cliques with more than three nodes is minimal and hence we removed clique indexing from Cheleby. After indexing the structures, Cheleby resolves corresponding unresponsive routers using GBI obeying the trace preservation condition.

Table 5.6 presents statistics for unresponsive router resolution steps. As indicated in Section 5.3.1, initial pruning resolves considerable number of unknown nodes. Then using indexing, we perform induction on the remaining ones to reduce the number of final unresponsive routers to 250K. This yields topologies where 17.24% of the routers are unresponsive, which agrees with our earlier observations regarding router responsiveness [61].

Table 5.6: Unresponsive Router Resolution (Average of 8 data sets)

Initial	I. Pruner	Rate Lim.	Triangle	Bipartite	Star	Final *s
7,207,885	6,137,750	51,279	2,858	143,880	619,204	252,915

Figure 5.16 presents the number of unresponsive router substrings with different lengths. For example, an unresponsive router substring with length 3 is a sub-trace of (a,*,*,*,b). As we expect, while the length of unresponsive routers increases, the number of unresponsive routers with that length increases.

Figure 5.17 presents the initial and final number of unresponsive routers. According to this figure, there are around 7M unresponsive routers in the raw data, while after the resolution this number is just around 250K, i.e., around 96% of the unresponsive nodes in the raw data is resolved. Additionally, in the final data %17.24 of the routers are unresponsive, which agrees with earlier observations [61].

Figure 5.18 presents the initial and final number of unresponsive routers for different unresponsive router lengths. According to this figure, while the substring length increases, the ratio of resolved unresponsive routers in this substring decreases. This is because in bipartite, and triangle resolution steps, we can just resolve the unresponsive router substrings with length 1. Thus, in longer substrings, the resolution ratio is lower.

Figure 5.19 shows the percentage of resolved unresponsive routers after each resolution step. In the star resolution step, we can resolve unresponsive router with any length. Hence, according to the table, the number of resolved unresponsive routers is the most in star resolution step. Since the number of triangle structures is low in datasets, the number of resolved unresponsive routers in the triangle step is small.

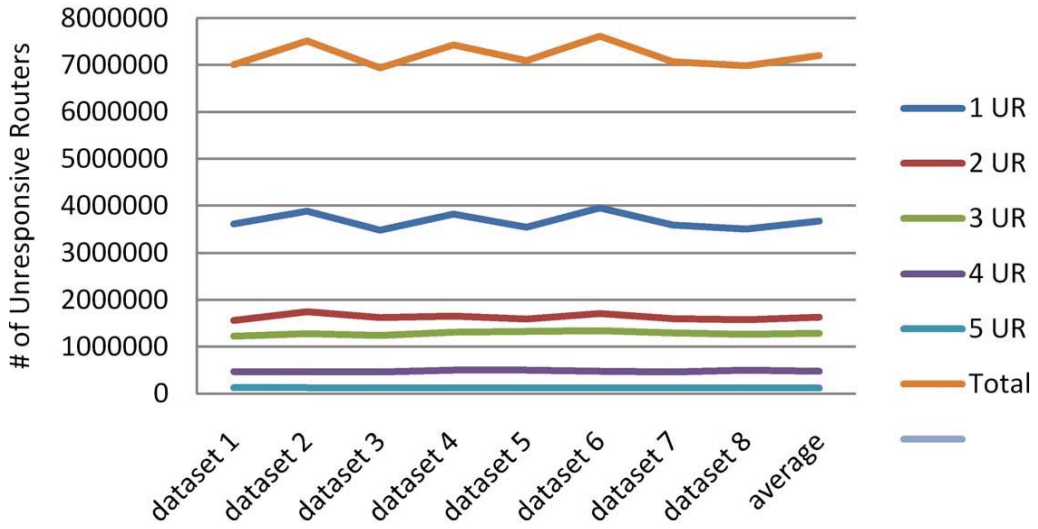


Figure 5.16: Number of Unresponsive Routers with Different Lengths

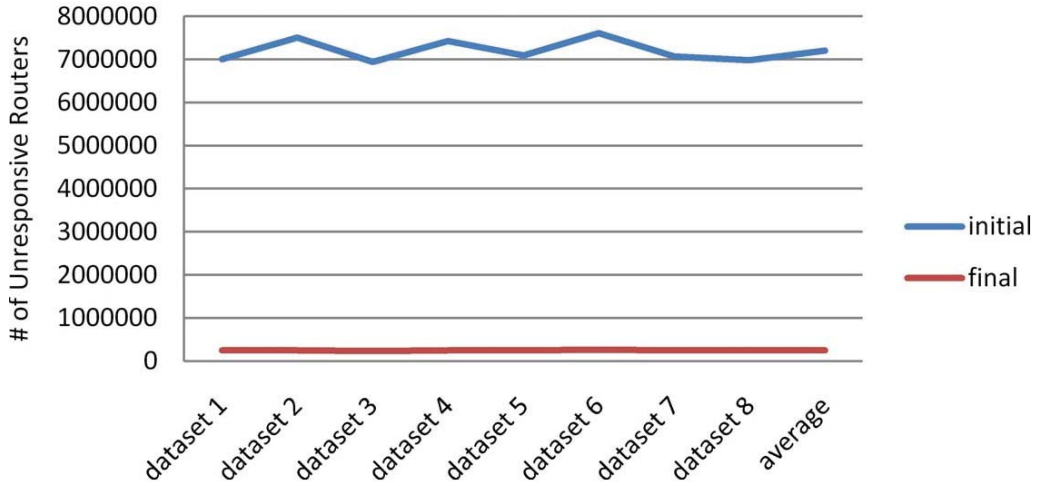


Figure 5.17: Initial vs. Final Unresponsive Routers

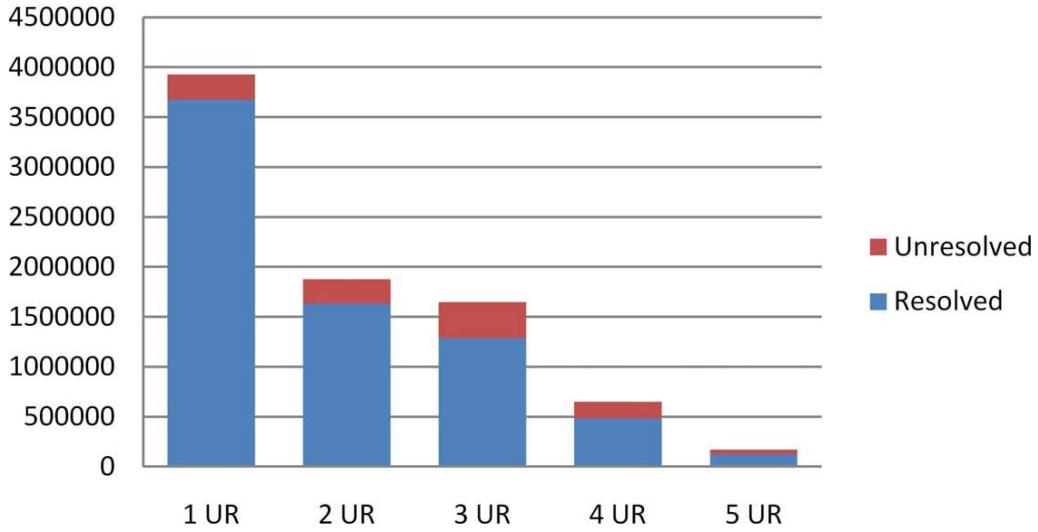


Figure 5.18: Initial vs. Final Unresponsive Routers according to UR length

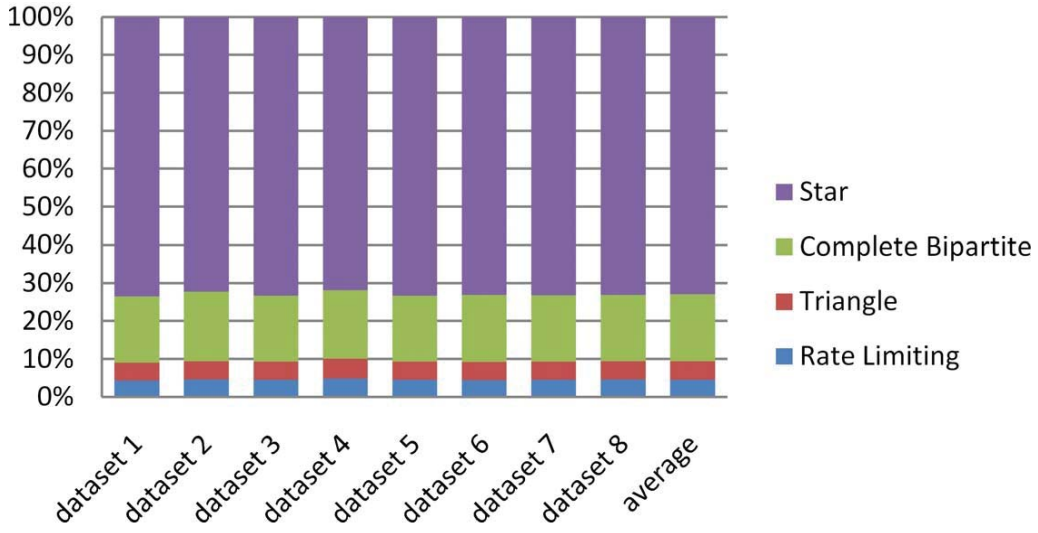


Figure 5.19: Percentages of Resolved Unresponsive Routers after Each Step

Table 5.7: Improved Subnet Statistics

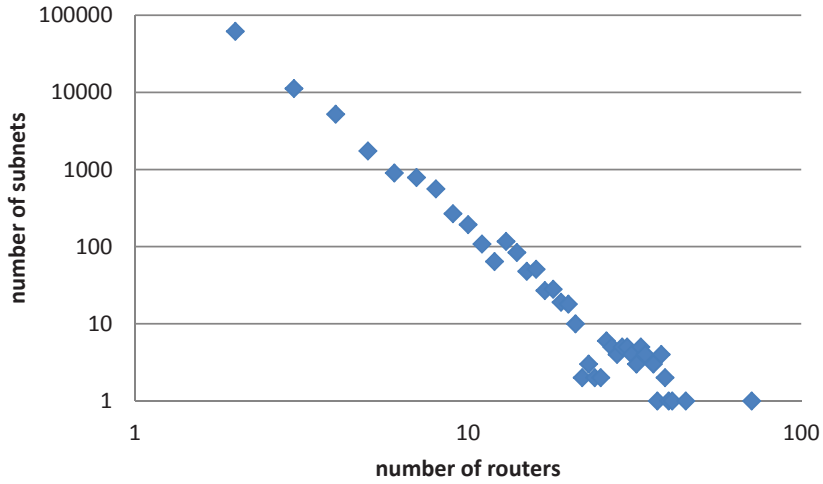
Subnet Size	/24	/25	/26	/27	/28	/29	/30	/31
Count	4	36	184	1,294	8,836	93,110	20,543	37,468
Completeness	26.3%	30.0%	28.3%	27.7%	28.0%	39.3%	100%	100%
All IPs	268	1,359	3,228	10,767	34,587	219,745	41,086	74,936

5.3.5 Increasing Graph Density

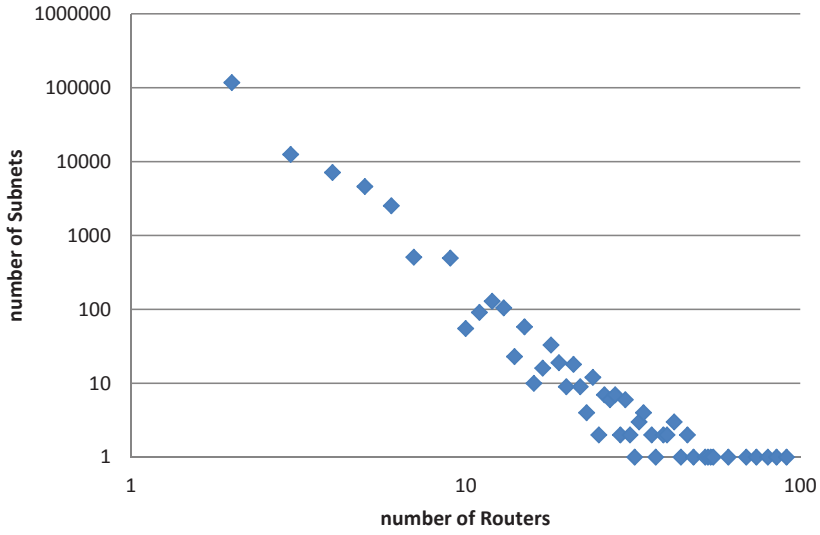
Realizing that many subnets had low completeness, we increased the probe destination coverage. For this, we determined non-observed IP addresses of candidate subnets that had at least 10% completeness. For instance, in a data-set there were 651.8K IP addresses missing from the identified candidate subnets. Moreover, we looked at /30 and /31 mate of observed IP addresses and they produced 535.2K and 93.1K IP addresses, respectively. Next, to ensure the existence of these IP addresses, we performed a reverse DNS lookup and probed them with a ping. If either of these tests were positive, we added them to the destination IP lists.

After these changes, we obtained a better resolution and a more complete topology. As seen in table 5.7, the number of observed subnets and their completeness significantly improved. In the final topology, the number of IP addresses observed in a subnet is about 400K, which is four times of the initial 99K. Additionally, Figure 5.20 presents the subnet size distribution of all observed subnets before and after increasing graph density.

Improvements in the subnet coverage and inclusion of IP-mates considerably improved alias IPs identified with APARv2 as seen in Table 5.8. The number of alias sets increased from 23K to 38K and the number of IP addresses in an alias set increased from 75K to 128K. Additionally, we implemented probing based mercator and ally approaches to complement APARv2. For mercator, we sent a probe to



a) Initial



b) After Increasing Graph Density

Figure 5.20: Cheleby Subnet Distribution

Table 5.8: Improved Alias Resolution Statistics

Resolver	Alias Sets	Aliased IPs
APARv2	38,012	128,495
Ally (path traces)	32,860	65,720
Ally (common neighbor)	32,595	65,190
Ally (subnet)	25,436	50,872
Ally (combined)	55,027	110,054
Mercator	305	610
Combined	82,962	216,628

all observed IP addresses and recorded the response. If the response was from an IP address different from the queried one, then we marked them as aliases. This approach produced the least number of aliases, i.e. only 610 IP addresses were added to an alias set.

Moreover, we utilized ally on candidate alias IP address pairs. For this, we identified candidates using three methods. (1) We identified path traces that had multiple IP addresses at a given hop distance. Then, we marked 70K IP address pairs at the same hop as candidate aliases to be probed with ally. (2) In the final graph, we identified IP addresses that had the same common neighbors, i.e., IP addresses whose neighbor intersection was more than one node. This produced 2M pairs of IP addresses as candidates to be probed. (3) We used subnets as pivot points to determine candidate aliases. For each subnet (e.g., consider subnet in Figure 2.3-a), we marked each subnet IP address (e.g., A, B, C and D) with the other IP addresses' neighboring IP addresses (i.e., A with the neighbors of {B, C, D}; B with the neighbors of {A, C, D}; C with the neighbors of {A, B, D}; and D with the neighbors of {A, B, C}). Subnet based candidate generation produced 3M candidate pairs to be probed. Probing these pairs with ally we identified aliases for 66K, 65K, and 51K IP addresses, respectively. Our results regarding traces with multiple IPs at a hop indicate that

majority of these IP addresses are actually aliases but not due to the load balancing. That is, among 70K IP addresses that appeared at the same hop 66K were actually aliases. After merging the resolved alias sets, we obtain 83K alias sets that contain 217K IP addresses, which is more than three times of the initial resolution results.

5.4 Summary

In this chapter, we present Cheleby, Internet topology mapping system that provides sample network topologies at the subnet layer. Cheleby is an assembly of state-of-the-art topology collection and construction techniques, i.e., target list generation, probe redundancy reduction, unbiased accurate data collection, subnet inference, alias resolution, and unresponsive router resolution, into a single system. The proposed Cheleby will enable research community to conduct topography analysis and study large-scale characteristics of the Internet as we publicly offer the resulting data sets.

Chapter 6

Internet Topology Mapping Systems

Several research groups have developed mapping systems to build Internet topology maps. Archipelago measurement infrastructure of CAIDA [2, 35, 85], the DIMES project [4, 110], and the iPlane infrastructure [6, 87] continuously provide sampled Internet topologies in order to facilitate topology measurement studies. Additionally, several other groups have developed various tools or systems [42, 51, 81, 91, 92, 103, 117, 122, 124].

In the rest of this chapter, we present major Internet topology mapping systems and several topological characteristics of topology maps provided by these systems.

6.1 Ark

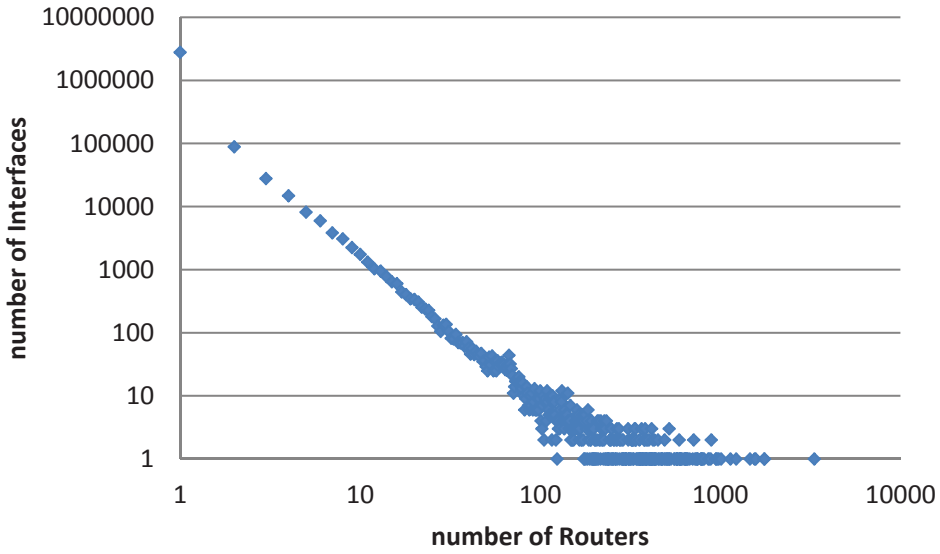
Archipelago [2] is a successor of the skitter measurement infrastructure [94] that started probing the Internet in 1998. A major step from Skitter to Ark is the coor-

dination of monitors using Marinda tuple-space, which utilizes a distributed memory space and pattern matching techniques [2]. Ark focuses on generating annotated Internet maps and currently utilizes 53 dedicated monitors around the world to trace every observed /24 subnetwork. Monitors are divided into 3 teams to trace towards 9.1M destination IP addresses using scamper [10] and generate approximately 100 probes per second. Ark started collecting IPv6 topology utilizing some of the monitors since September 2010.

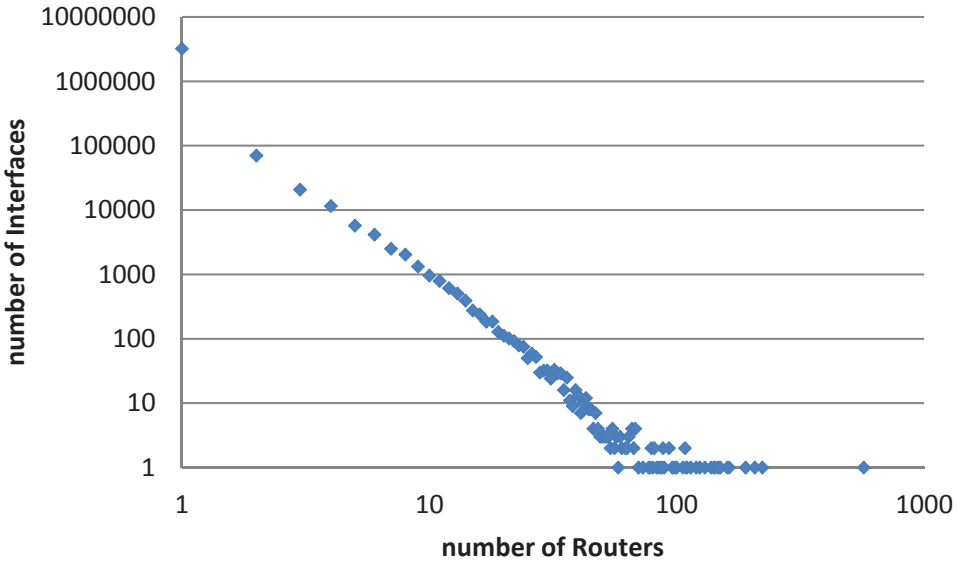
Ark utilizes midar [80], kapar [78], and iffinder [5] tools to resolve IP aliases [79]. They publicly provide two different router interface lists. For the first list, they use iffinder, midar and kapar tools to resolve IP aliases. For the second list, they do the alias resolution on a map by just utilizing the iffinder and midar tools. Figure 6.1-a shows the interface distribution of the routers for the first list, while Figure 6.1-b represents the interface distribution of the routers for the second list. Additionally, Figure 6.2 presents the node degree distribution for topology map provided by Ark.

6.2 Dimes

Similar to SETI@home crowd sourcing approach [17], Distributed Internet Measurements and Simulations (DIMES) [110] utilizes home computers to collect path traces around the world. Currently, around 20K agents around the world contribute as vantage points to probe destinations from a rich set of locations and capture peripheral Internet topology. DIMES focuses on PoP level topology mapping and annotating the links with the delay and loss statistics. Finally, DIMES only implements Mercator method in resolving IP aliases.



a) Using kapar, midar, and iffinder)



b) Using midar, ad iffinder

Figure 6.1: Ark Router Interface Distribution

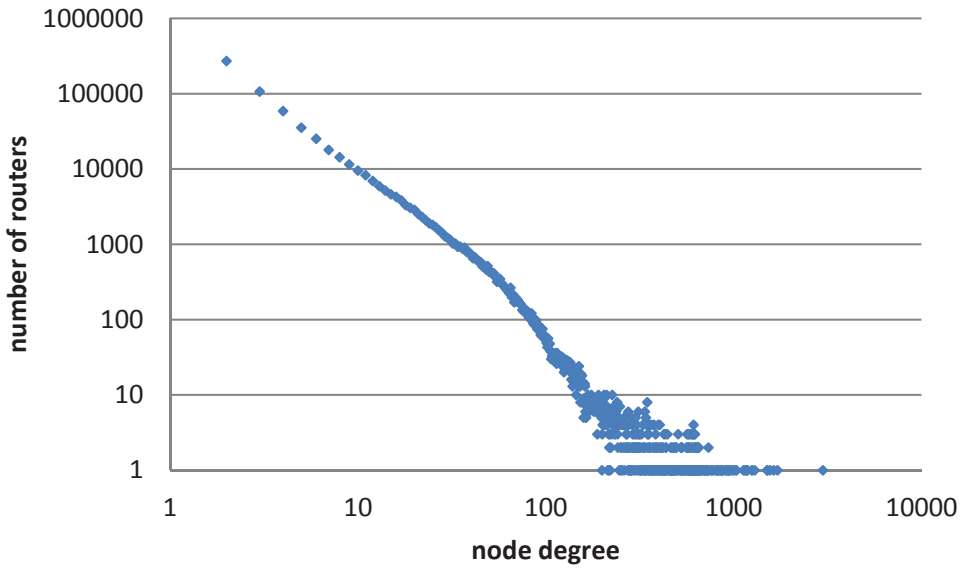
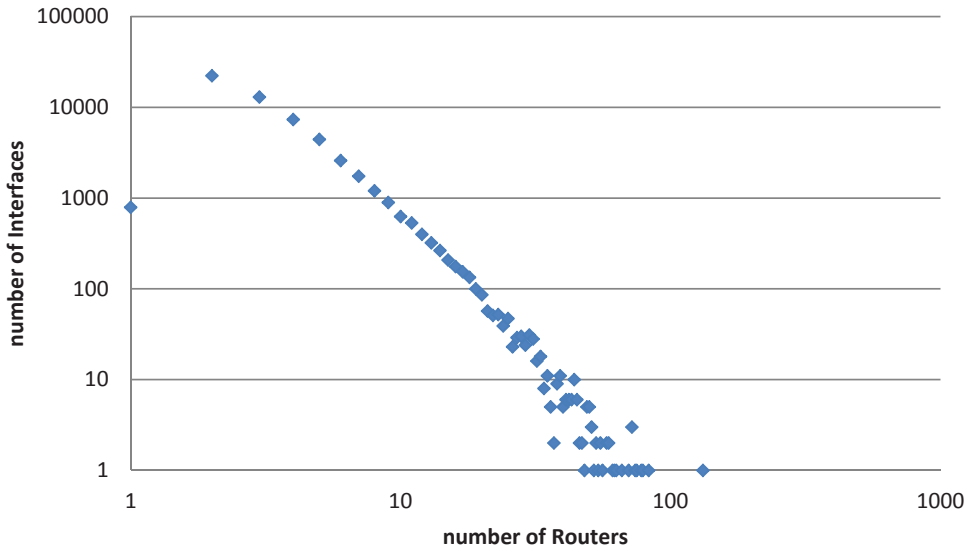


Figure 6.2: Ark Node Degree Distribution

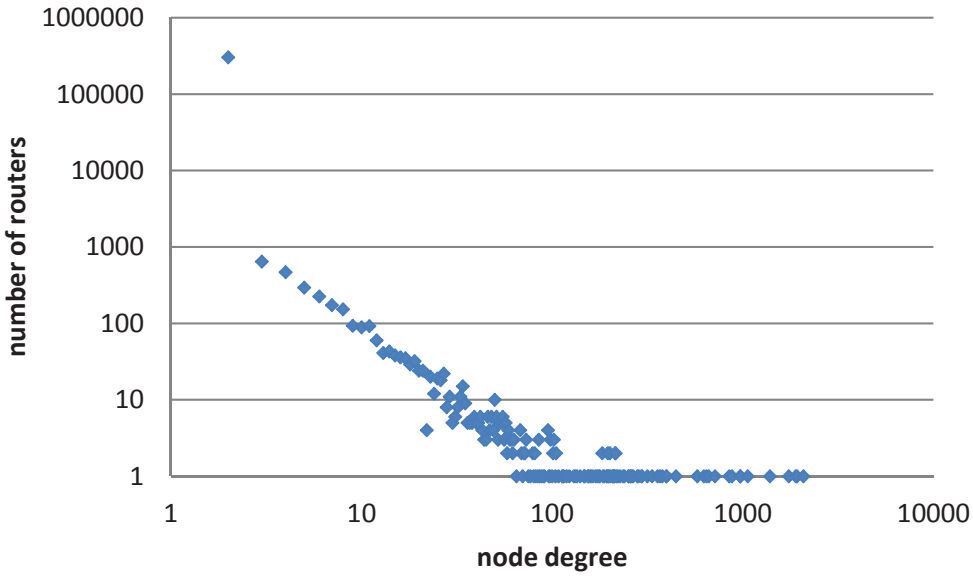
6.3 iPlane

iPlane [6] aims at providing Internet links annotated with the latency, bandwidth, capacity and loss rate for improved overlay network deployment. iPlane performs path traces from 200 PlanetLab monitors towards 100K destinations to construct a backbone topology that can be used as landmarks for the overlay networks. Moreover, the geo-location of routers is identified using the undns [114] and sarangworld [9] tools. Finally, iPlane utilizes Mercator and Ally in resolving IP aliases.

iPlane provides a list of IP aliases in its website [7]. Figure 6.3-a presents the interface distribution of the routers for a topology map collected by iPlane infrastructure while Figure 6.3-b represents the node degree distribution for the same topology.



a) Router Interface Distribution)



b) Node Degree Distribution

Figure 6.3: iPlane characteristics

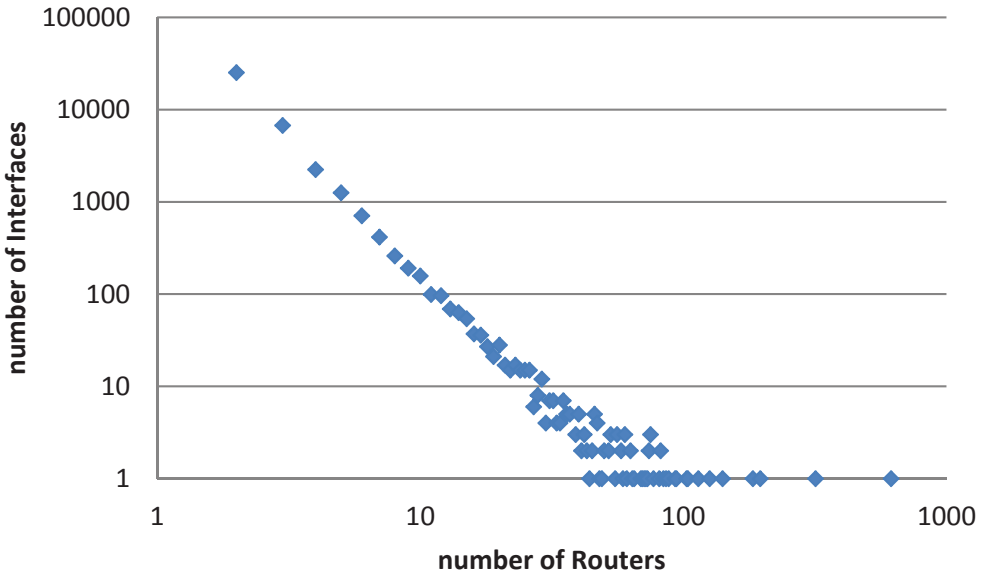
6.4 Cheleby

As mentioned above, there are several systems providing the raw router-level Internet topology data, but there is no system providing the up-to-date constructed topologies. This is simply because of the fact that the literature on topology construction is fairly limited compared to the studies in topology collection and topology analysis. Moreover, topology construction is not a straightforward process requiring considerable computations.

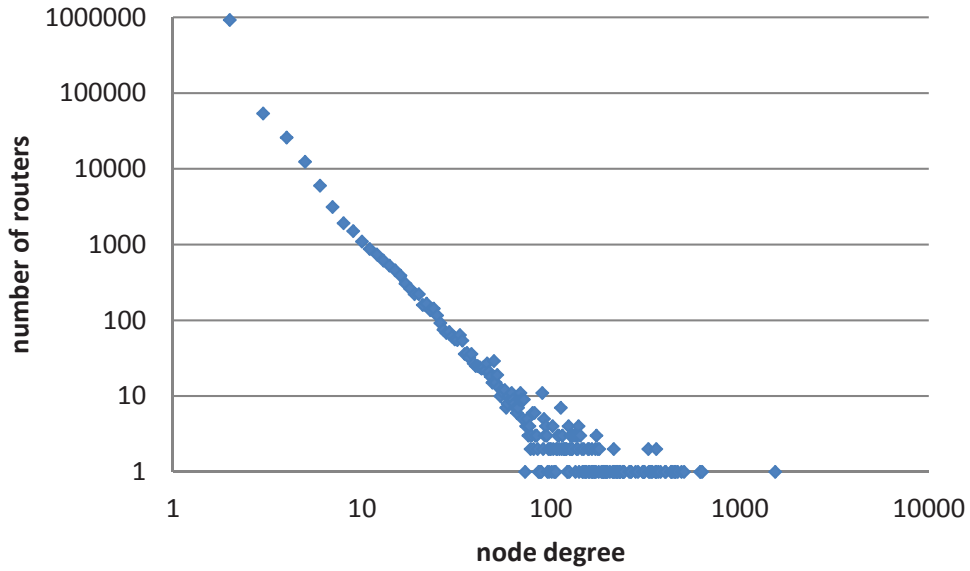
Inaccuracies in the topology sampling and construction processes may significantly affect the accuracy of the observations or results obtained in the measurement study [16, 27, 53, 58, 84, 118]. However, currently deployed topology mapping systems do not complete all topology construction tasks. In particular, they provide alias pairs for some data sets but ignore unresponsive routers and subnets of observed IP addresses. Addition of subnet relations and resolving unresponsive routers in the final graph would considerably improve the accuracy of sample topologies.

On the other hand, Cheleby first dynamically probes observed subnetworks using a team of PlanetLab nodes around the world to obtain comprehensive topology data. Then, it utilizes efficient algorithms for resolving subnets, IP aliases, and unresponsive routers in collected data sets to build accurate subnet-level topologies. Incorporating enhanced resolution algorithms, Cheleby provides comprehensive topology maps. Unlike previous approaches, Cheleby provides both the raw and the constructed Internet topology data.

Figure 6.4-a presents the interface distribution of the routers for a topology map collected by Cheleby infrastructure while Figure 6.4-b represents the node degree distribution for the same topology.



a) Router Interface Distribution)



b) Node Degree Distribution

Figure 6.4: Cheleby characteristics

Table 6.1: Internet Topology Mapping Systems

	Ark [2]	DIMES [4]	iPlane [6]	Cheleby [3]
Monitors	53	19K	200	500
Dest. IP	9.1M		100K	3.5M
Traces	27.1M	3.6M	33.8M	22M
Probes	993M	15M	472M	658M
IPs	1.3M		0.3M	1.2M
Edges	2.3M		1.2M	2.9M
Alias Sets	79.6K		12.1K	83K
Aliased IPs	271K		33.2	217K
Data Type	Router/AS	Router/PoP/AS	PoP/AS	Subnet-level

Table 6.1 presents major Internet topology mapping systems and their characteristics including the number of: (1) deployed monitors, (2) destination IP addresses, (3) collected traces, (4) generated probes, (5) observed IP addresses, (6) observed edges without topology construction, (7) alias sets, (8) IP addresses that appeared in an alias set, and (9) provided data type[1]. Note that, iPlane sends a single probe per hop during path tracing while other systems send three. Using three probes per hop helps the mapping system identify the load balancing routers and carefully infer the links between hops.

[1]As DIMES does not release raw traces we could not obtain some of its statistics.

Chapter 7

Conclusions

Due to the tremendous growth in Internet's importance, many groups, organizations, and governments have become interested in understanding various characteristics of the Internet for commercial, social, and technical reasons. Network research community depends on such Internet mapping systems to understand characteristics of the Internet and develop better protocols and services. Government agencies are interested in Internet measurements to protect and improve the national cyber infrastructure. Moreover, new network paradigms such as overlay networks require knowledge of the underlying topology.

In this dissertation, we first present the Structural Graph Indexing (SGI) for efficiently mining complex networks. As indexing feature, we utilize graph structures such as star, complete bipartite, triangle and clique that frequently appear in protein, chemical compound, and Internet graphs. SGI lists all substructures matching structure formulations and new structures can be identified and added to the SGI. Different from previous approaches, SGI does not limit the number of nodes in the indexes and provides an alternative tool for querying large graphs. In evaluation of the

SGI, we perform experiments on genuine Internet topology and Wikipedia datasets to demonstrate the viability of this algorithm for large datasets.

In order to assess the extent of unresponsive routers in Internet topology mapping studies, we first present an experimental study on the responsiveness of routers to active probe messages. In our historical analysis, we observed that responsiveness reduced during the last decade. We also observed that network operators are increasingly rate limiting ICMP responses. Another observation from our study is that the destination reachability considerably reduced over the time indicating that systems (i.e., routers and end systems) are increasingly unwilling to respond to direct probes. We also observed that routers are less willing to respond to direct active probes as compared to indirect active probes. Finally, our experiments showed that the responsiveness of routers changes with the type of the probes; ICMP based probes eliciting the highest response rate and UDP based ones eliciting the lowest. Even though TCP based probes receive responses better than UDP based ones, they are more likely to raise security alerts.

We then enhance the *Graph Based Induction* (GBI) unresponsive router resolution approach by incorporating our SGI technique and utilizing the findings from our measurement study. In SGI, we index observed subgraphs that contain unknown nodes and determine the corresponding minimal underlying structure that satisfies the trace accuracy condition. We also show that proper resolution of IP aliases improves the unresponsive resolution. Our work improves the state of the art in unresponsive router resolution in terms of accuracy and efficiency. Regarding accuracy, GBI addresses the lack of (i) *temporary unresponsive router resolution* as the number of temporarily unresponsive routers have been increasing and (ii) **-subpath resolution* as the length of *-subpaths and the number of longer ones have been increasing.

Neither of these cases are properly addressed by previous resolution studies while both appear increasingly often in the recent years. Regarding efficiency, the run time complexity of our algorithm is significantly less than that of the existing algorithms. Our experiments on three different data sets have shown a significant reduction in the practical run time overhead of SGI (approximately, $5.2*10^7$ operations) as compared to the previous approaches (approximately, 10^{12}, 10^{18}, or 10^{30} operations), in the worst case.

Finally, we present Cheleby Internet topology mapping system that provides sample network topologies at the subnet layer (available at `cheleby.cse.unr.edu`). Cheleby assembles state-of-the-art topology collection and construction techniques, i.e., target list generation, probe redundancy reduction, unbiased accurate data collection, subnet inference, alias resolution, and unresponsive router resolution. Cheleby will enable research community to conduct topography analysis and study large-scale characteristics of the Internet as we publicly provide the resulting data sets.

Overall, the Cheleby Internet mapping system helps to (i) provide a finer grade Internet topologies at subnet level, (ii) better understand the characteristics of the Internet topology, (iii) capture the dynamics of Internet backbone, (iv) fine-tune existing services such as content distribution or bottleneck identification, (v) and guide the development of the next generation Internet.

Bibliography

[1] *ally tool*. http://www.cs.washington.edu/research/networking/rocketfuel/.

[2] *Archipelago Measurement Infrastructure (Ark)*. http://www.caida.org/projects/ark.

[3] *Cheleby: An Internet Topology Mapping System*. http://cheleby.cse.unr.edu/.

[4] *Dimes Project*. http://www.netdimes.org.

[5] *iffinder tool*. http://www.caida.org/tools/measurement/iffinder/.

[6] *iPlane*. http://iplane.cs.washington.edu/.

[7] *iPlane Alias Cluster List*. http://iplane.cs.washington.edu/data/alias_lists.txt.

[8] *PlanetLab Project*. http://www.planet-lab.org.

[9] *Sarangworld project*. http://www.sarangworld.com/TRACEROUTE/.

[10] *Scamper*. http://www.wand.net.nz/scamper/.

[11] Route selection in cisco routers. Technical Report Document ID: 8651, Cisco, January 2008. available at http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094823.shtml.

[12] Configuring a load-balancing scheme. Technical report, Cisco, October 2011. available at http://www.cisco.com/en/US/docs/ios-xml/ios/ipswitch_cef/configuration/15-0m/isw-cef-load-balancing.pdf.

[13] N. M. A. Thomas, R. Cannings and C. Cannings. On the structure of protein-protein interaction networks. *Biochem. Soc. Trans.*, 31:1491 – 1496, 2003.

[14] H. B. Acharya and M. G. Gouda. A theory of network tracing. In *11th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 62–74, Berlin, Heidelberg, 2009. Springer-Verlag.

[15] A. Almog, J. Goldberger, and Y. Shavitt. Unifying unknown nodes in the internet graph using semisupervised spectral clustering. In *Proceedings of the 2008 IEEE International Conference on Data Mining Workshops*, pages 174–183, Washington, DC, USA, 2008. IEEE Computer Society.

[16] L. Amini, A. Shaikh, and H. Schulzrinne. Issues with inferring Internet topological attributes. In *Proceedings of SPIE ITCom*, Boston, MA, USA, July/August 2002.

[17] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. Seti@home: an experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, 2002.

[18] M. Antic and A. Smiljanic. Routing with load balancing: increasing the guaranteed node traffics. *Communications Letters, IEEE*, 13(6):450 –452, june 2009.

[19] G. Antichi, A. Di Pietro, D. Ficara, S. Giordano, G. Procissi, and F. Vitucci. Network topology discovery through self-constrained decisions. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1 –6, nov. 2009.

[20] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with Paris traceroute. In *Proceedings of IMC*, Rio de Janeiro, Brazil, October 2006.

[21] B. Augustin, F. Friedman, and R. Teixeira. Measuring loadbalanced paths in the Internet. In *Proceedings of IMC*, San Diego, CA, October 2007.

[22] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '02, pages 623–632, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[23] P. Barford, A. Bestavros, J. Byers, and M. Crovella. On the marginal utility of network topology measurements. In *ACM Internet Measurements Workshop*, San Francisco, CA, USA, November 2001.

[24] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient algorithms for large-scale local triangle counting. *ACM Trans. Knowl. Discov. Data*, 4(3):13:1–13:28, Oct. 2010.

[25] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns, 1999.

[26] A. Bender, R. Sherwood, and N. Spring. Fixing ally's growing pains with velocity modeling. In *8th ACM SIGCOMM IMC*, pages 337–342, New York, NY, USA, 2008. ACM.

[27] S. Bilir, K. Sarac, and T. Korkmaz. Intersection characteristics of end-to-end Internet paths and trees. In *IEEE International Conference on Network Protocols (ICNP)*, Boston, MA, USA, November 2005.

[28] A. Broido and K. Claffy. Internet topology: Connectivity of IP graphs. In *Proceedings of SPIE ITCom Conference*, Denver, CO, USA, August 2001.

[29] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, 1973.

[30] H. Chang, S. Jamin, and W. Willinger. Inferring as-level Internet topology from router-level path traces. In *Proceedings of SPIE ITCom*, Denver, CO, August 2001.

[31] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. The origin of power laws in Internet topologies revisited. In *Proceedings of IEEE INFOCOM*, New York, NY, USA, June 2002.

[32] L. Cheng, N. Hutchinson, and M. Ito. Realnet: A topology generator based on real internet topology. In *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, pages 526 –532, 25-28 2008.

[33] B. Cheswick, H. Burch, and S. Branigan. Mapping and visualizing the Internet. In *ACM USENIX*, San Diego, CA, USA, June 2000.

[34] C.-W. Chung, J.-K. Min, and K. Shim. Apex: an adaptive path index for xml data. In *SIGMOD Conference*, pages 121–132, 2002.

[35] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov. Internet mapping: From art to science. *Conference For Homeland Security, Cybersecurity Applications & Technology*, 0:205–211, 2009.

[36] J. Cong and M. Smith. A parallel bottom-up clustering algorithm with applications to circuit partitioning in vlsi design. In *DAC '93: Proceedings of the 30th international Design Automation Conference*, pages 755–760, New York, NY, USA, 1993. ACM.

[37] D. Cook and L. Holder. *Mining graph data*. John Wiley & Sons, 2006.

[38] B. F. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon. A fast index for semistructured data. In *VLDB*, pages 341–350, 2001.

[39] A. Coyle, M. Kraetzl, O. Maennel, and M. Roughan. On the predictive power of shortest-path weight inference. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 305–310, New York, NY, USA, 2008. ACM.

[40] CYMRU. http://www.team-cymru.org/services/ip-to-asn.html.

[41] B. Donnet, B. Huffaker, T. Friedman, and K. Claffy. Increasing the coverage of a cooperative internet topology discovery algorithm. In *Proceedings of IFIP NETWORKING*, Atlanta, GA, USA, May 2007.

[42] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Efficient algorithms for large-scale topology discovery. In *Proceedings of ACM/SIGMETRICS*, pages 327–338, New York, NY, USA, 2005. ACM Press.

[43] J.-P. Eckmann and E. Moses. Curvature of co-links uncovers hidden thematic layers in the world wide web. *Proceedings of the National Academy of Sciences of the United States of America*, 99(9):5825–5829, Apr. 2002.

[44] B. Eriksson, G. Dasarathy, P. Barford, and R. Nowak. Toward the practical use of network tomography for internet topology discovery. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, 14-19 2010.

[45] D. Fay, H. Haddadi, A. Thomason, A. Moore, R. Mortier, A. Jamakovic, S. Uhlig, and M. Rio. Weighted spectral distribution for internet topology analysis: Theory and applications. *Networking, IEEE/ACM Transactions on*, 18(1):164 –176, feb. 2010.

[46] D. Feldman and Y. Shavitt. An optimal median calculation algorithm for estimating Internet link delays from active measurements. In *IEEE E2EMON*, Munich, Germany, May 2007.

[47] G. Fertin, A. Raspaud, and B. A. Reed. On star coloring of graphs. In *Proceedings of the 27th International Workshop on Graph-Theoretic Concepts in Computer Science*, WG '01, pages 140–153, London, UK, UK, 2001. Springer-Verlag.

[48] S. Garcia-Jimenez, E. Magana, D. Morato, and M. Izal. Techniques for better alias resolution in internet topology discovery. In *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, pages 513 –520, 1-5 2009.

[49] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[50] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. The flattening internet topology: natural evolution, unsightly barnacles or contrived collapse? In *PAM'08: Proceedings of the 9th international conference on Passive and active network measurement*, pages 1–10, Berlin, Heidelberg, 2008. Springer-Verlag.

[51] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. In *IEEE INFOCOM*, Tel Aviv, ISRAEL, March 2000.

[52] J. Guillaume and M. Latapy. Relevance of massively distributed explorations of the Internet topology: Simulation results. In *Proceedings of IEEE INFOCOM*, Miami, FL, USA, March 2005.

[53] M. Gunes, S. Bilir, K. Sarac, and T. Korkmaz. A measurement study on overhead distribution of value-added Internet services. *Computer Networks*, 51(14):4153–4173, 2007.

[54] M. Gunes and K. Sarac. Analytical IP alias resolution. In *IEEE International Conference on Communications (ICC)*, Istanbul, TURKEY, June 2006.

[55] M. Gunes and K. Sarac. Quantifying the impact of alias resolution on traceroute-based sample network topologies. Technical report, University of Texas at Dallas, September 2006.

[56] M. Gunes and K. Sarac. Resolving IP aliases in building traceroute-based Internet maps. Technical report, University of Texas at Dallas, December 2006.

[57] M. Gunes and K. Sarac. Impact of alias resolution on traceroute-based sample network topologies. In *Passive and Active Measurement Conference (PAM)*, Louvain-la-neuve, BELGIUM, April 2007.

[58] M. Gunes and K. Sarac. Importance of IP alias resolution in sampling Internet topologies. In *IEEE Global Internet (GI)*, Anchorage, AK, May 2007.

[59] M. Gunes and K. Sarac. Inferring subnets in router-level topology collection studies. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, San Diego, CA, October 2007.

[60] M. Gunes and K. Sarac. Resolving anonymous routers in Internet topology measurement studies. In *Proceedings of IEEE INFOCOM*, Phoenix, AZ, USA, April 2008.

[61] M. H. Gunes and K. Sarac. Analyzing router responsiveness to active measurement probes. In *Proceedings of the 10th International Conference on Passive and Active Network Measurement*, PAM '09, pages 23–32, Berlin, Heidelberg, 2009. Springer-Verlag.

[62] M. H. Gunes and K. Sarac. Resolving ip aliases in building traceroute-based internet maps. *IEEE/ACM Trans. Netw.*, 17:1738–1751, December 2009.

[63] H. Haddadi, D. Fay, S. Uhlig, A. Moore, R. Mortier, and A. Jamakovic. Mixing biases: Structural changes in the AS topology evolution. In *TMA '10: Proceedings of the Second International Workshop on Traffic Measurements and Analysis*. Springer, April 2010.

[64] H. Haddadi, M. Rio, G. Iannaccone, A. Moore, and R. Mortier. Network topologies: inference, modeling, and generation. *Communications Surveys Tutorials, IEEE*, 10(2):48 –69, second 2008.

[65] S. L. Hakimi, J. Mitchem, and E. Schmeichel. Star arboricity of graphs. *Discrete Math.*, 149(1-3):93–98, Feb. 1996.

[66] Y. He, G. Siganos, M. Faloutsos, and S. Krishnamurthy. Lord of the links: A framework for discovering missing links in the internet topology. *Networking, IEEE/ACM Transactions on*, 17(2):391 –404, april 2009.

[67] V. Jacobson. *Traceroute*. Lawrence Berkeley Laboratory (LBL), February 1989. Available from ftp://ee.lbl.gov/traceroute.tar.Z.

[68] C. A. James, D. Weininger, and J. Delany. Daylight theory manual - daylight 4.91, Apr. 2005.

[69] X. Jin, W. Tu, and S.-H. Chan. Scalable and efficient end-to-end network topology inference. *Parallel and Distributed Systems, IEEE Transactions on*, 19(6):837 –850, june 2008.

[70] X. Jin, W.-P. K. Yiu, S.-H. G. Chan, and Y. Wang. Network topology inference based on end-to-end measurements. *IEEE Journal on Selected Areas in Communications special issue on Sampling the Internet*, 24(12):2182–2195, Dec. 2006.

[71] D. Johnson, D. Gebhardt, and J. Lepreau. Towards a high quality path-oriented network measurement and storage system. In *PAM'08: Proceedings of the 9th international conference on Passive and active network measurement*, pages 102–111, Berlin, Heidelberg, 2008. Springer-Verlag.

[72] S. Kandula and R. Mahajan. Sampling biases in network path measurements and what to do about it. In *Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference*, pages 156–169, New York, NY, USA, 2009.

[73] H. Kardes, M. Gunes, and T. Oz. Cheleby: A subnet-level internet topology mapping system. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1 –10, jan. 2012.

[74] H. Kardes and M. H. Gunes. Structural graph indexing for mining complex networks. In *Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems Workshops*, ICDCSW '10, pages 99–104, Washington, DC, USA, 2010. IEEE Computer Society.

[75] H. Kardes, M. H. Gunes, and T. Oz. Cheleby: A subnet-level internet topology mapping system. In *COMSNETS*, pages 1–10, 2012.

[76] E. Katz-Bassett, H. Madhyastha, J. John, A. Krishnamurthy, D. Wetherall, and T. Anderson. Studying black holes in the Internet with hubble. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*, San Fransicso, CA, USA, April 2008.

[77] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for indexing paths in graph-structured data. In *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, pages 129–140, Washington, DC, USA, 2002. IEEE Computer Society.

[78] K. Keys. Ip alias resolution techniques - version 1.1. Technical report, Cooperative Association for Internet Data Analysis - CAIDA, Dec 2008. (2009-01-19).

[79] K. Keys. Internet-scale IP alias resolution techniques. *SIGCOMM Comput. Commun. Rev.*, 40:50–55, January 2010.

[80] K. Keys, Y. Hyun, M. Luckie, and K. Claffy. Internet-scale ipv4 alias resolution with midar: System architecture. Technical report, Cooperative Association for Internet Data Analysis - CAIDA, May 2011.

[81] S. Kim and K. Harfoush. Efficient estimation of more detailed Internet IP maps. In *IEEE International Conference on Communications (ICC)*, Glasgow, Scotland, June 2007.

[82] M. Kwon and S. Fahmy. Topology-aware overlay networks for group communication. In *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 127–136, New York, NY, USA, 2002. ACM.

[83] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In S. Kalyanaraman, V. N. Padmanabhan, K. K. Ramakrishnan, R. Shorey, and G. M. Voelker, editors, *Proceedings of the ACM SIGCOMM 2010 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New Delhi, India, August 30 -September 3, 2010*, pages 75–86. ACM, 2010.

[84] A. Lakhina, J. Byers, M. Crovella, and P. Xie. Sampling biases in IP topology measurements. In *IEEE INFOCOM*, San Francisco, CA, USA, March 2003.

[85] M. Luckie, Y. Hyun, and B. Huffaker. Traceroute probe method and forward ip path inference. In *8th ACM SIGCOMM IMC*, pages 311–324, New York, NY, USA, 2008. ACM.

[86] M. Luckie, Y. Hyun, and B. Huffaker. Traceroute Probe Method and Forward IP Path Inference. In *Internet Measurement Conference (IMC)*, pages 311–324, Vouliagmeni, Greece, Oct 2008. Internet Measurement Conference (IMC).

[87] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *OSDI*, November 2006.

[88] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, k. c. claffy, and A. Vahdat. The internet as-level topology: three data sources and one definitive metric. *SIGCOMM Comput. Commun. Rev.*, 36(1):17–26, 2006.

[89] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring link weights using end-to-end measurements. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 231–236, New York, NY, USA, 2002. ACM.

[90] T. Matsuda, H. Motoda, and T. Washio. Graph-based induction and its applications. *Advanced Engineering Informatics*, 16(2):135–1434, April 2002.

[91] A. McGregor, H.-W. Braun, and J. Brown. The NLANR network analysis infrastructure. *IEEE Communications Magazine*, 38(5):122–128, May 2000.

[92] T. McGregor, S. Alcock, and D. Karrenberg. The RIPE NCC Internet Measurement Data Repository. In *11th PAM*, April 2010.

[93] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[94] D. McRobb, K. Claffy, and T. Monk. *Skitter: CAIDA's macroscopic Internet topology discovery and tracking tool*, 1999. Available from http://www.caida.org/tools/skitter/.

[95] T. Milo and D. Suciu. Index structures for path expressions. In *ICDT*, volume 1540 of *Lecture Notes in Computer Science*, pages 277–295. Springer, 1999.

[96] K. Naidu, D. Panigrahi, and R. Rastogi. Detecting anomalies using end-to-end path measurements. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1849 –1857, 13-18 2008.

[97] A. Nakao, L. Peterson, and A. Bavier. A routing underlay for overlay networks. In *Proceedings of ACM SIGCOMM*, pages 11–18, Karlsruhe, Germany, August 2003.

[98] M. Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[99] M. E. J. Newman. The structure and function of complex networks. *SIAM REVIEW*, 45:167–256, 2003.

[100] J. Ni, H. Xie, S. Tatikonda, and Y. Yang. Efficient and dynamic routing topology inference from end-to-end measurements. *Networking, IEEE/ACM Transactions on*, 18(1):123 –135, feb. 2010.

[101] J. Pansiot and D. Grad. On routes and multicast trees in the Internet. *ACM Computer Communication Review*, 28(1), January 1998.

[102] R. Pastor-Satorras and A. Vespignani. *Evolution and Structure of the Internet*. Cambridge University Press, 2004.

[103] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis. An architecture for large-scale Internet measurement. *IEEE Communications*, August 1998.

[104] R. Peeters. The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics*, 131(3):651 – 654, 2003.

[105] N. Rhodes, P. Willett, A. Calvet, J. B. Dunbar, and C. Humblet. Clip: similarity searching of 3d databases using clique detection. *J Chem Inf Comput Sci.*, 43(2):443 – 448, 2003.

[106] N. Robertson and P. D. Seymour. Graph minors: X. obstructions to tree-decomposition. *J. Comb. Theory Ser. B*, 52(2):153–190, June 1991.

[107] J. M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3):425 – 440, 1986.

[108] R. Samudrala and J. Moult. A graph-theoretic algorithm for comparative modeling of protein structure. *Journal of Molecular Biology*, 279(1):287 – 302, 1998.

[109] D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *In Symposium on Principles of Database Systems*, pages 39–52, 2002.

[110] Y. Shavitt and E. Shir. DIMES: Let the Internet measure itself. *SIGCOMM Comput. Commun. Rev.*, 35(5):71–74, 2005.

[111] Y. Shavitt and U. Weinsberg. Quantifying the importance of vantage points distribution in internet topology measurements. In *IEEE INFOCOM 2009*, pages 792 –800, 19-25 2009.

[112] R. Sherwood, A. Bender, and N. Spring. Discarte: a disjunctive internet cartographer. In *Proceedings of the ACM SIGCOMM 2008*, pages 303–314, New York, NY, USA, 2008. ACM.

[113] R. Sherwood and N. Spring. Touring the Internet in a TCP sidecar. In *Proceedings of the ACM/SIGCOMM on Internet Measurement Conference*, pages 339–344, New York, NY, USA, 2006. ACM Press.

[114] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies using rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, Feb 2004.

[115] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies using rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, February 2004.

[116] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A public internet measurement facility. In *Proceedings of USENIX Symposium on Internet Technologies and Systems*, March 2003.

[117] N. Spring, D. Wetherall, and T. Anderson. Reverse engineering the internet. *SIGCOMM Comput. Commun. Rev.*, 34(1):3–8, 2004.

[118] R. Teixeira, K. Marzullo, S. Savage, and G. Voelker. In search of path diversity in ISP networks. In *Proceedings of the USENIX/ACM Internet Measurement Conference*, Miami, FL, USA, October 2003.

[119] M. E. Tozal and K. Sarac. Tracenet: an internet topology data collector. In *Proceedings of the 10th Annual conference on Internet measurement*, IMC '10, pages 356–368, New York, NY, USA, 2010. ACM.

[120] M. E. Tozal and K. Sarac. Palmtree: An ip alias resolution algorithm with linear probing complexity. *Computer Communications*, 34(5):658 – 669, 2011. Special Issue: Complex Networks.

[121] M. E. Tozal and K. Sarac. Subnet level network topology mapping. In *Proceedings of the 30th IEEE International Performance Computing and Communications Conference*, IPCCC '11, 2011.

[122] S. Triukose, Z. Wen, A. Derewecki, and M. Rabinovich. Dipzoom: An open ecosystem for network measurements. In *Proceedings of IEEE INFOCOM*, Anchorage, AK, USA, May 2007.

[123] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos. Doulion: counting triangles in massive graphs with a coin. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 837–846, New York, NY, USA, 2009. ACM.

[124] University of Oregon. Route views project. http://www.routeviews.org.

[125] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*. Cambridge Univ Pr, 1994.

[126] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. pages 255–270, Boston, MA, Dec. 2002.

[127] X. Yan, P. S. Yu, and J. Han. Graph indexing: A frequent structure-based approach. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 335–346, New York, NY, USA, 2004. ACM.

[128] B. Yao, R. Viswanathan, F. Chang, and D. Waddington. Topology inference in the presence of anonymous routers. In *IEEE INFOCOM*, San Francisco, CA, USA, March 2003.

[129] K. Yoshida, Y. Kikuchi, M. Yamamoto, Y. Fujii, K. Nagami, I. Nakagawa, and H. Esaki. Inferring pop-level isp topology through end-to-end delay measurement. In *PAM '09: Proceedings of the 10th International Conference on Passive and Active Network Measurement*, pages 35–44, Berlin, Heidelberg, 2009. Springer-Verlag.

[130] E. W. Zegura, K. L. Calvert, and M. J. Donahoo. A quantitative comparison of graph-based models for Internet topology. *IEEE/ACM Transactions on Networking*, 5(6):770–783, 1997.

[131] Y. Zhao, Y. Chen, and D. Bindel. Towards unbiased end-to-end network diagnosis. *Networking, IEEE/ACM Transactions on*, 17(6):1724 –1737, dec. 2009.