ABSTRACT

Toward a Robust Internet Interdomain Routing

by
Jian Wu

Chair: Kang G. Shin

Robustness has always been one of the most important requirements in the design of the Internet infrastructure. This dissertation takes two directions toward enhancing the robustness of today's Internet interdomain routing. On one hand, we propose *reactive* techniques to identify the cause and origin of each routing instability after its occurrence. On the other hand, we develop a *proactive* mechanism to enable the current interdomain routing protocol to tolerate certain types of failures.

We first focus on the analysis of BGP dynamics from a single network's perspective and develop a troubleshooting system that identifies in real-time from millions of daily BGP updates a few routing events that network operators can take direct actions upon to alleviate their impacts.

There is serious lack of understanding of Internet routing resilience to significant and realistic failures such as those caused by the 2003 Northeast Blackout and the 2006 Taiwan earthquake. We systematically analyze how the current Internet routing system reacts to various types of failures by developing a realistic failure model, and then use it to pinpoint the reliability bottlenecks of the Internet. By focusing on the impacts of structural and policy properties, our analysis provides guidelines for future Internet design.

We find that the current policy-driven interdomain routing greatly limits the Internet's ability to maintain normal reachability under adverse conditions, and therefore, propose *dynamic routing negotiation* (DRN) to allow ISPs to temporarily relax routing policy restrictions when needed, to exploit the existing physical redundancy in the network topology.

The increasing security concerns and emerging MPLS-like layer-2 technology make the traditional tools such as traceroute less capable of identifying the internal structure of networks, which is very important to diagnosis of network anomalies. To reduce the opaqueness of today's networks, we propose a novel approach to discovering the resource sharing of each network based on the performance measurement between each pair of ingress and egress points in the network. Its performance and utility have been demonstrated via extensive simulations and Internet experiments.

Toward a Robust Internet Interdomain Routing

by

Jian Wu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2009

Doctoral Committee:
Professor Kang G. Shin, Chair
Professor Farnam Jahanian
Assistant Professor Zhuoqing Morley Mao
Assistant Professor Clayton D. Scott

To Mom, Dad, and Shirley

ACKNOWLEDGEMENTS

I had never foreseen to have such a long and bumpy journey. So many times have I found myself stuck in the middle of nowhere, befuddled at the directions where my research can proceed. Thanks to the tremendous guidance and support from many wonderful people surrounding me, I have finally been able to complete this dissertation.

First and foremost, I would like to thank my advisor, Prof. Kang G. Shin, for his continuous guidance and encouragement throughout these years. He gave me a great deal of freedom and patience to explore possible research ideas and I have benefited a lot from his technical insights, in particular, the perseverance that I have found essential for any basic research effort.

I am very fortunate to be given the opportunity to work with Prof. Z. Morley Mao. She has been a fantastic mentor. She introduced me to the field of Internet routing and been involved throughout this dissertation research. I owe a great debt to her for the patience and inspirational guidance for the past four years. I would also like to thank Prof. Farnam Jahanian and Prof. Clayton Scott for serving on my thesis committee for their incredible insight and advice that helped me to improve my work.

A special thanks goes to Ying Zhang, who has given me tremendous help during most of my thesis work. My research has also benefited from many colleagues in RTCL. I thank Daji Qiao, Haining Wang, Wei Sun, Zhigang Chen, Chun-Ting Chou, Chang-Hao Tsai for their friendship, collaborative work and constructive discussions. I also thank Mohamed El Gendy, Kyu-Han Kim, Hyoil Kim as well as other members in the networking group for their invaluable comments on my research work.

My parents have always been my strongest supporters and encouraged me to pull through and move forward. I also thank my wife, Shirley, for her care and encouragement. I dedicate this dissertation to them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
Introduction

Despite the widespread use of the Internet and its impact on practically every segment of our society, its workings remain poorly understood by most users. Nevertheless, more and more users take it for granted to be able to boot up their laptops anywhere (e.g., cafes, airports, hotels) and connect to the Internet to use services such as email, web browsing, or even watching the trailers of the latest movies. The few times the users get a glimpse of the complexities of the infrastructure that supports such ubiquitous communication are when they experience various “networking” problems (e.g., the familiar “cannot connect” message, or unacceptably poor performance), because diagnosing such problems typically exposes certain aspects of the underlying network architecture (how the components of the network infrastructure interrelate) and network protocols (standards that govern the exchange of data). The Internet’s success and popularity is to a large degree due to its ability to hide most of its complexity and gives users the illusion of a single, seamlessly connected network where the fragmented nature of the underlying infrastructure and the many layers of protocols remain largely transparent to the users. However, the fact that the Internet is, in general, very successful in hiding from the user most of the underlying details and intricacies does not make them go away. In fact, even Internet experts admit having more and more troubles getting (and keeping) their arms around the essential components of this large-scale, highly-engineered network that has all the features typically associated with complex systems.

1.1 Design Principles of the Internet Architecture

When viewed in terms of its hardware, the Internet consists of hosts or end points (also called end systems), routers or internal switching stations (also referred to as gateways), and links that connect the various hosts and/or routers and can differ widely in speed (from slow modem connection to high-speed backbone links) as well as in technology. When viewed from the perspective of *autonomous systems* (ASes), where an AS is a collection of routers and links under a single administrative domain (e.g., a company, an organization, or a school), the network is an internetwork consisting of a number of separate subnetworks or ASes, interlinked to give users the illusion of a single, seamlessly connected network (network of networks, or "Internet"). A network architecture is a framework that aims at specifying how the different components of the networks interrelate. More precisely, a "*network architecture is a set of high-level design principles that guides the technical design of a network, especially the engineering of its protocols and algorithms. It sets a sense of direction – providing coherent and consistency to the technical decisions that have to be made and ensuring that certain requirement are met*" [1].

Much of what we refer to as today's Internet is the result of an architectural network design that was developed in the 1970s under the auspices of the Defense Advanced Research Project Agency (DARPA) of the US Department of Defense. The main objective of the original DARPA Internet architecture was inter-networking – the development of an "effective techniques for multiplexed utilization of already existing interconnected (but typically separately administered) networks." A set of objectives, originally published in [2], essentially elaborates on the meaning of the word "effective" and defines a more detailed list of goals for the original Internet architecture. These requirements are (in decreasing order of importance):

- *Robustness*: Internet communications must continue despite loss of networks or gateways/routers.
- *Heterogeneity*: The Internet must support multiple types of communication services and the Internet architecture must accommodate a variety of networks.
- *Distributed Management*: The Internet architecture must permit distributed management of its resources.
- *Cost*: The Internet architecture must be cost-effective.

- *Ease of Attachment*: The Internet architecture must permit host attachment with a small amount of effort

- *Accountability*: The resources used in the Internet architecture must be accountable.

This priority-ordered list of requirements, first and foremost among them, the robustness criterion, has to a large degree been responsible for shaping the architectural model and the design of the protocols (standards governing the exchange of data) that define today's Internet. In particular, "robustness" means to provide the Internet some underlying capability in the presence of uncertainty. That is, the Internet must be (1) flexible to changes in technology, use of the network; (2) able to maintain continuous service in the face of failures.

1.2 Robustness in Internet Routing

In the five-layer TCP/IP protocol stack used in the Internet, IP (Internet Protocol) layer manages to ensure that any packet anywhere in the network is forwarded to the correct next hop until the destination is reached. Addressing and routing are crucial aspects that enable IP to achieve this task. Each device in the Internet has a unique address that it uses to label its network interface. Each packet generated by any of these devices has source and destination addresses, where the former references the local interface address and the latter gives the corresponding interface address of the intended recipient of the packet. When handing packets over from one router to another within the network, each router is able to identify the intended receiver of each packet. Maintaining sufficient and consistent information within the network for associating the identity of the intended recipient with its location inside the network is achieved by means of *routing protocols*; that is, a set of distributed algorithms that the routers run among themselves to make appropriate routing decisions. The routing protocol is designed so that each router can not only identify a set of output interfaces that can be used to move a packet closer to its destination, but also select an interface which represents the best possible path to that destination. Robustness considerations that play a role in this context include randomly occurring router or link failures and restoration of failed network components or the addition of new components to the network.

The design of techniques that ensures routing resilience to failures in the physical infrastructure of the Internet can be divided into two manageable pieces, where the division is in accordance with

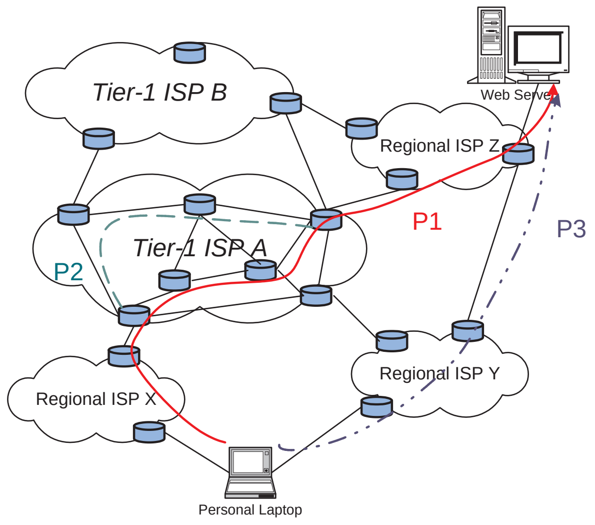

Figure 1.1: Diagram of Internet routing

separation of the Internet into ASes for improved scalability: each AS runs a local internal routing protocol (or Interior Gateway Protocol (IGP)), and between the different ASes, an internetwork routing protocol (or Exterior Gateway Protocol (EGP)) maintains connectivity and ties all of the ASes together and ensures seamless communication across AS boundaries. For simplicity, we call them *intradomain* routing protocols and *interdomain* routing protocols, respectively. As shown in Figure 1.1, inside ISP A, the intradomain routing protocol decides to choose whether to take path $P1$ or path $P2$, while the interdomain routing dictates choice between path $P1$ and path $P3$. In this thesis, we focus on the design and operation of the Border Gateway Protocol (BGP), the de-facto standard interdomain routing protocol deployed in today's Internet.

BGP [3] is a "path vector" routing protocol that constructs paths by successively propagating updates between pairs of BGP-speaking routers that establish BGP peering sessions. Each BGP update concerns a particular prefix and includes the list of ASes along the paths to reach the destination from the BGP speaker. Each BGP-speaking router originates updates for one or more prefixes, and can send the updates to its immediate neighbors via BGP sessions. Upon receipt of BGP updates, the routers perform a routing decision process to determine the best route for each destination prefix among the routes learned from its neighboring routers. The simplest path-vector protocol would employ the shortest AS path routing, where each AS selects a route with the shortest AS path. However, BGP allows a much wider range of routing policies so as to honor contractual agreements between ASes that control the exchange of traffic. This feature enables an administratively decentralized Internet: by using these policies, ASes can direct traffic to ASes with whom they have business relationships, where traditional network routing protocols would have selected the shortest path. BGP is an incremental routing protocol. As the network undergoes changes (e.g., link failures, provisioning of a new link), BGP uses advertisement and withdrawal messages to inform neighboring routers of the routing changes. An advertisement indicates that a certain path to a given destination is used and a withdrawal notifies that a previously advertised path to a destination is no longer available.

In summary, BGP uses distributed computation and relies on the exchanges of updated routing information to maintain consistent knowledge across different ASes to ensure the seamless communications in the Internet. The distributed nature of BGP path selection inevitably raises concerns and requires special attention to potential problems with routing instability (i.e., oscillations) and slow convergence, which are common to many distributed routing protocols. For example, to rate-limit advertisements, BGP uses timers associated with the Minimum Route Advertisement Internet (MRAI) parameter. When a BGP-speaking router sends a route advertisement for a given destination to a neighbor, it starts an instance of this timer. The router is not allowed to send another advertisement concerning this destination to that neighbor until the corresponding timer has expired. While waiting for the MRAI timer to expire, the router in question may receive many updates for the same destination and can privately enumerate many alternative choices of its best path without burdening its neighbor with the ephemeral intermediate updates. Using the MRAI timer reduces the number of updates needed for convergence but adds some delay to the whole convergence process. Overall, the BGP specifications explicitly mention five timers. In general, determining default values for these timers has been mostly guess work, and little is known about their effects on the dynamics of BGP in today's Internet.

1.3 Pitfalls in BGP

The current interdomain routing protocol, BGP, has evolved over the past decade and now constitutes a critical part of the Internet infrastructure. The substantial complexity of BGP mainly comes from the need to support flexible policies while scaling to a large number of Autonomous

Systems (ASes).

- *Policy*: ASes have business relationship with each other to satisfy their respective financial goals, and in the meantime, must cooperate to achieve global reachability. Operators use routing policies to control the flow of traffic and specify which routes are advertised to neighboring networks under what conditions.
- *Scalability*: Routing protocols must scale with increasing network size. The main mechanism to achieve scalability is aggressive aggregation of routing information. For example, BGP abstracts an AS as a single node. Each BGP route contains the sequence of ASes rather than routers that advertised the route.

To get a glimpse of the pitfalls inherent in BGP, we consider the following two problems. The first problem is due to the policy that each AS independently enforces. Previous work found that BGP is vulnerable to persistent oscillations, such as the "bad gadget" scenario [4]. In this situation, three (or more) ASes continually oscillate between their available routing choices because each AS prefers to route indirectly via another AS rather than directly to the destination. This type of BGP oscillation is essentially resulted from BGP's freedom in specifying policies and inability to satisfy group preferences. The second problem comes from the side-effect of BGP's information hiding to achieve scalability. The abstraction of an AS as a single node makes BGP to scale, but they also make it difficult to determine the cause and the origin of a routing update because an AS has essentially no information about the origin of a route change or withdrawal (often at router level). The inability to pinpoint the source of a routing update slows convergence and complicates problem diagnosis. More foundational problems are posed and extensively discussed in [5]. This research attempts to tackle some of the pitfalls in BGP and improve the robustness of the Internet interdomain routing.

1.4 Challenges and Contributions

Internet routing is dynamic in nature. Caused by the regular or irregular exchange of routing updates between routers, routing dynamics have always been a major concern of the Internet engineering community. Irregular dynamics can not only cause high bandwidth and processing overhead on routers, but may also lead to poor end-to-end performance, caused by packet losses, delays, delay jitters, reordering, etc. Understanding routing dynamics allows us to pinpoint network anomalies and pathologies, identify potential protocol or router design defects, and suggest better designs of next-generation Internet routing protocols. Despite the extensive studies of Internet routing dynamics, especially BGP dynamics [6, 7, 8] in the last few years, they are still poorly understood. In his work on developing a signal propagation model for BGP updates, Griffin [9] said “In practice, BGP updates are perplexing and interpretation is very difficult”.

Traditional BGP root-cause analysis [10, 11, 12, 13, 14] aim to identify the origin and the cause of each routing event that is responsible for the propagation of BGP updates. Unfortunately, due to the complexity of the BGP dynamics and the insufficient information of the topology and routing policies in each AS, the inference achieved from these studies is often inaccurate. In Chapter 2, we focus on the BGP updates viewed from a single AS and develop a tool which identifies from millions of daily BGP updates the few routing events that network operators can take direct actions upon to alleviate their impacts. Instead of attempting to account for each of the routing events, the design principle in the tool is to capture what is the most interesting to the network operators and the users of each network.

Typical events that cause network link or router failures include accidental cable/fiber cuts, hardware malfunctions, power outage, software bugs, natural disasters (e.g., fire or earthquake), human errors (e.g., misconfigurations, incorrect maintenance/upgrade), or even terrorist attacks, Denial-of-Service (DOS) attacks. As evidence of just how frequently failures occur, Snow [15] has reported that since 1992 there have been about 16 outages per month in the United States alone that each affected over 30,000 users. Interesting (even bizarre) reports [16, 17, 18] of cable cuts and their impacts can also be found daily on the Internet. Certain failures, due to its large scale, tend to have a more significant impact on the connectivity of the Internet. The robustness of the Internet routing is thus critical under extreme conditions, failures such as the 911 terrorist attack [19], the 2003 Northeast Blackout [20], and the Taiwan Earthquake in 2006 [21]. In Chapter 3, we propose a framework to systematically analyze how the current Internet routing system reacts to various types of failures. In particular, our technique is shown to be able to pinpoint the reliability bottlenecks of the Internet.

As described in Section 1.2, interdomain (i.e., BGP) routing is policy-driven. Because of the

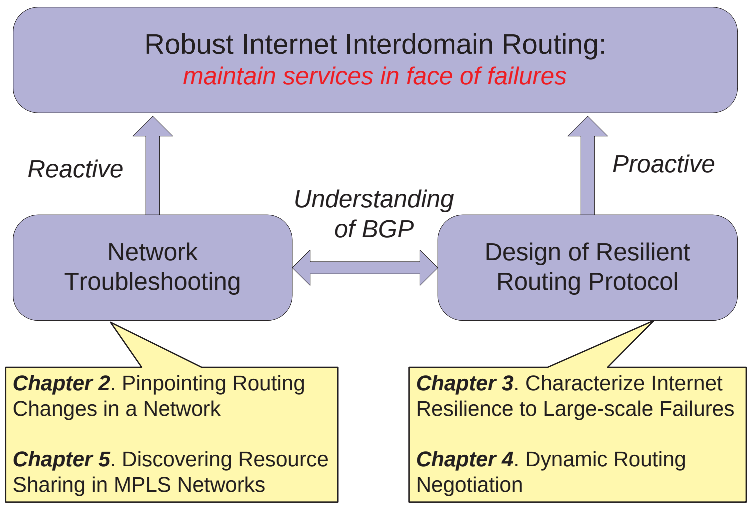
Robust Internet Interdomain Routing:
maintain services in face of failures
Reactive
Proactive
Understanding of BGP
Network Troubleshooting
Design of Resilient Routing Protocol
Chapter 2. Pinpointing Routing Changes in a Network
Chapter 5. Discovering Resource Sharing in MPLS Networks
Chapter 3. Characterize Internet Resilience to Large-scale Failures
Chapter 4. Dynamic Routing Negotiation

Figure 1.2: Roadmap of the dissertation

policy restrictions imposed by each individual AS based on its business relationships with its neighbors, physical connectivity does not directly translate into reachability. The impact of the restrictions on the robustness of the interdomain routing is magnified when certain failures significantly cripple the Internet routing, as identified in Chapter 3. In Chapter 4, we propose a novel idea of dynamic routing negotiations to allow ISPs to temporarily relax policy restrictions when needed, to enhance Internet routing robustness by better utilizing the existing physical redundancy in the network topology.

The knowledge of network topology can always be beneficial to diagnosing network anomalies and devising measures to alleviate their effects, as manifested by the fact that traditional measurement tools like traceroute are indispensable for operators to do network troubleshooting. Unfortunately, due to the increasing concerns on network security, compounded by the recent emergence of MPLS-like layer-2 technology, these tools become less capable of identifying the internal structure of the Internet. To reduce the opaque nature of today's Internet, in Chapter 5, we propose a novel approach to discover the internal structure of each network based on the performance measurement obtained between each pair of ingress and egress points in the network.

Figure 1.2 illustrates the roadmap of this dissertation. In this research, we enhance the robustness of interdomain routing both *reactively* and *proactively*. On one hand, through the work in Chapter 2 and Chapter 5, we develop techniques to provide more intelligent network troubleshooting in the face of failures. On the other hand, the work in Chapter 3 and Chapter 4 equips the interdomain routing protocol with more inherent features to evade from the effects of Internet failures. Lastly, Chapter 6 summarizes the main contributions of the dissertation and suggests possible directions of future work.

CHAPTER 2

Pinpointing Significant BGP Routing Changes in an IP Network

This dissertation begins with an effort to tackle Internet routing robustness in a *reactive* fashion. The intent is to provide a network troubleshooting tool to identify the origin and cause of large routing disruptions *after* they occur so that mitigations can be applied accordingly to alleviate the impacts of these disruptions and enhance the routing robustness.

2.1 Introduction

Ensuring good performance in an IP backbone network requires continuous monitoring to detect and diagnose problems, as well as quick responses from management systems and human operators to limit the effects on end users. Network operators need to know when destinations become unreachable to notify affected customers and track down the cause of the problem. When measurements indicate that links have become congested, operators may respond by modifying the routing protocol configurations to direct some traffic to other lightly-loaded paths. These kinds of measurements are also crucial for discovering weaknesses in existing network protocols, router implementations, and operational practices to drive improvements for the future. All of these tasks require effective ways to cull through large amounts of measurement data, often in real time, to produce concise, meaningful reports about changes in network conditions.

To track events inside their own network, operators collect measurements of data traffic, performance statistics, the internal topology, and equipment failures. The performance of a backbone network is especially vulnerable to *interdomain* routing changes that affect how data traffic travels to destinations in other Autonomous Systems (ASes). For example, a link failure in a remote AS could trigger a shift in how traffic travels through a network, perhaps causing congestion on one or more links. Fortunately, operators can gain additional visibility into the interdomain routing changes by monitoring the Border Gateway Protocol (BGP) decisions of routers at the periphery of their AS. In this chapter, we address the challenge of analyzing a large volume of BGP update messages from multiple routers in real time to produce a small number of meaningful alerts for the operators.

In addition to the large volume of data, producing useful reports is challenging because: (i) BGP update messages show the changes in AS-level paths without indicating why or where they originated, (ii) a single network event (such as a failure) can lead to multiple update messages during routing protocol convergence, (iii) a single network event may affect routing decisions at multiple border routers, and (iv) a single event may affect multiple destination prefixes. Having a small number of reports that highlight only *important* routing changes is crucial to avoid overwhelming the operators with too much information. The reports should focus on routing changes that disrupt reachability, generate a large number of update messages, affect a large volume of traffic, or are long-lived enough to warrant corrective action. These concerns drive the design of our system. We have evaluated our system on two months of data from a tier-1 ISP and discovered several important problems that were previously unknown. Our system analyzes millions of BGP update messages per day to produce a few dozen actionable reports for the network operators.

Despite some high-level similarities, our approach differs markedly from recent work on root-cause analysis of BGP routing changes [10, 11, 12, 13, 14]. These studies analyze streams of BGP update messages from vantage points throughout the Internet, with the goal of inferring the location and cause of routing changes. Instead, we consider BGP routing changes seen *inside* a single AS to identify—and quantify—the *effects* on that network. Realizing that root-cause analysis of routing changes is intrinsically difficult [22], we search only for explanations of events that occur close to the AS—such as internal routing changes and the failure of BGP sessions with neighboring domains—and mainly focus on alerting operators to the performance problems they can address. Hence, our approach is complementary to previous work on root-cause analysis, while producing results of direct and immediate use to network operators.

In the next section, we present background material on BGP, followed by an overview of our system in Section 2.3. In Section 2.4, we group BGP update messages into routing *events*. We identify persistently flapping prefixes and pinpoint the causes. In Section 2.5, we introduce the concept of a *route vector* that captures the best BGP route for each prefix at each border router. We identify five types of routing changes that vary in their impact on the traffic flow. In Section 2.6 we group events by type to identify frequently flapping prefixes, BGP session resets, and internal routing disruptions; we validate our results using RouteViews data, syslog reports, and intradomain topology data. In Section 2.7, we use prefix-level traffic measurements to estimate the impact of the routing changes. Section 2.8 shows that our system operates quickly enough to generate reports in real time. Section 2.9 presents related work, and Section 2.10 concludes the chapter.

2.2 BGP Overview

The Border Gateway Protocol (BGP) [3] is the routing protocol that ASes use to exchange information about how to reach destination address blocks (or *prefixes*). Three key aspects of BGP are important for our study:

Path-vector protocol: Each BGP advertisement includes the list of ASes along the path, along with other attributes such as the next-hop IP address. By representing the path at the AS level, BGP hides the details of the topology and routing inside each network.

Incremental protocol: A router sends an advertisement of a new route for a prefix or a withdrawal when the route is no longer available. Every BGP update message is indicative of a routing change, such as the old route disappearing or the new route becoming available.

Policy-oriented protocol: Routers can apply complex policies to influence the selection of the best route for each prefix and to decide whether to propagate this route to neighbors. Knowing why a routing change occurs requires understanding how policy affected the decision.

To select a single best route for each prefix, a router applies the decision process [3] in Table 2.1 to compare the routes learned from BGP neighbors. In backbone networks, the selection of BGP routes depends on the interaction between three routing protocols:

External BGP (eBGP): The border routers at the periphery of the network learn how to reach external destinations through eBGP sessions with routers in other ASes. A large network often has multiple eBGP sessions with another AS at different routers. This is a common requirement for two ASes to have a peering relationship, and even some customers connect in multiple locations

1. Ignore if the next hop is unreachable;
2. Highest local preference;
3. Shortest AS path;
4. Lowest origin type;
5. Lowest Multiple-Exit-Discriminator (MED) value among routes from same AS;
6. eBGP routes over iBGP routes;
7. Lowest IGP cost (“hot-potato”);
8. Lowest router ID;

Table 2.1: BGP decision process

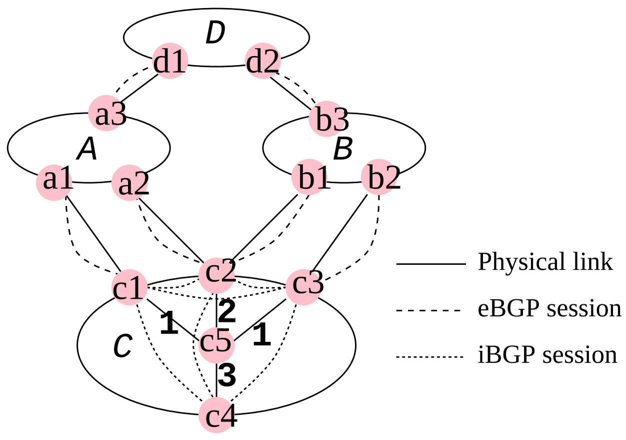

Figure 2.1: Interaction of routing protocols in AS C

for enhanced reliability. For example, Figure 2.1 shows AS C has two eBGP sessions with AS A and two eBGP sessions with AS B. As a result, there are three egress points to destinations in AS D.

Internal BGP (iBGP): After applying local policies to the eBGP-learned routes, a border router selects a single best route and uses iBGP to advertise the route to the rest of the AS. In the simplest case, each router has an iBGP session with every other router (*i.e.*, a *full-mesh* iBGP configuration). In Figure 2.1, the router $c4$ learns a two-hop AS path to destinations in AS D from three routers $c1$, $c2$, and $c3$.

Interior Gateway Protocol (IGP): The routers inside the AS run IGP to learn how to reach each other. The two most common IGPs are OSPF and IS-IS, which compute shortest paths based on configurable link weights. The routers use the IGP path costs in the seventh step in Table 2.1 to

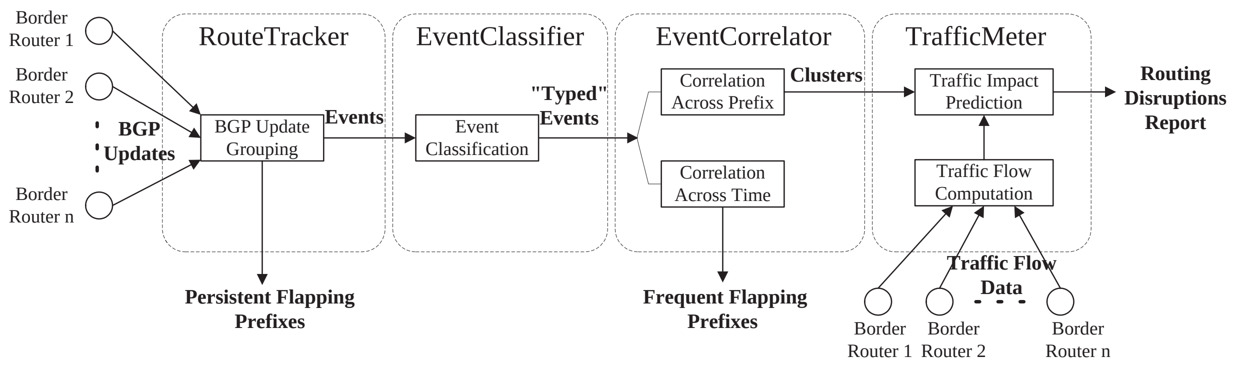

Figure 2.2: System design

select the *closest* egress point. In Figure 2.1, the number near each link inside AS C indicates the IGP cost of the link. Based on the decision rules, $c4$ prefers the routes through $c1$ and $c3$ over the route through $c2$ due to the smaller IGP path costs.[1]

The decision process in Table 2.1 allows us to compare two routes based on their attributes. We exploit this observation to determine whether a router switched from a better route to a worse route, or vice versa.

2.3 System Architecture

In this section, we describe how to track the BGP routing changes in an AS. Then, we present an overview of our system and describe the data we collected from a Tier-1 ISP backbone to demonstrate the utility of our tool.

2.3.1 Measurement Infrastructure

The routers at the edge of an AS learn BGP routes via eBGP sessions with neighboring domains, and then send update messages to other routers inside the AS via iBGP sessions. These border routers have complete visibility into external and internal routing changes. Ideally, each border router would provide a complete, up-to-date view of *all* routes learned from eBGP and iBGP neighbors. This data would allow our system to emulate the BGP decision process of each router, to understand why a router switched from one BGP route to another. Unfortunately, acquiring a timely feed of all eBGP updates received from neighboring ASes is difficult in practice.[2]

In this study, we analyze routing changes using only the data readily available in today's networks—a feed of the *best* route for each prefix from each border router. Our monitor has an iBGP session with each border router to track changes to the best route over time. A daily snapshot of the routing table from each border router is also collected to learn the initial best route for each prefix.

Since routing changes can have a significant effect on the distribution of the traffic over the network, traffic measurements are very useful for quantifying the *impact* of a routing change. In our measurement infrastructure, the monitor receives a feed of prefix-level traffic statistics from each border router. Because our analysis focuses on how routing changes affect the way traffic *leaves* the network, we collect the outgoing traffic on the edge links emanating from the border routers.

2.3.2 System Components

Our troubleshooting system analyzes BGP routing changes visible from inside a single AS and quantifies the effects on the network. The system is designed to operate *online* so operators may take corrective actions to improve network performance. For ease of presentation, we describe the functionality of our system in four distinct stages, as illustrated in Figure 2.2:

RouteTracker (Section 2.4): The first module merges the streams of BGP updates from the border routers and identifies routing *events*—groups of update messages for the same prefix that occur close in time. Along the way, the module identifies prefixes that flap continuously.

EventClassifier (Section 2.5): The second module classifies the routing events in terms of the kind of routing change and the resulting impact on the flow of traffic through the network. For example, we define a category called *internal disruption* that pinpoints the events caused by internal topology changes.

EventCorrelator (Section 2.6): The third module identifies related events by clustering over time and prefixes. In contrast to previous studies [10, 11, 12, 13, 14], we focus mainly on events that occur very close to the network (*e.g.,* eBGP session resets or internal disruptions) and have a significant impact on traffic. In addition, our correlation algorithms consider whether the border routers switched from a better route to a worse route, or vice versa—information not readily available in eBGP data feeds used in previous work on BGP root-cause analysis.

Component	**Reduction**	**Factor**
RouteTracker	updates $\rightarrow$ events	15.2
EventCorrelator	events $\rightarrow$ clusters	31.7
TrafficMeter	clusters $\rightarrow$ “important” clusters	327.6
Total	updates $\rightarrow$ “important” clusters	158460

Table 2.2: Incremental information reduction

TrafficMeter (Section 2.7): The last module estimates the impact of routing changes on the flow of traffic, to draw the operators’ attention to the most significant traffic shifts. Using prefix-level measurements of the traffic leaving the network, TrafficMeter computes a traffic weight that estimates the relative popularity of each prefix. The module predicts the severity of each event cluster by adding the weights of the affected prefixes.

In moving from raw updates to concise reports, we apply time windows to combine related updates and events, and thresholds to flag clusters with significant traffic volumes. We use our measurement data and an understanding of BGP dynamics to identify appropriate time windows; the threshold values reflect a trade-off between the number and significance of the disruptions we report.

2.3.3 Applying the System in a Tier-1 ISP

We have applied our prototype to a Tier-1 ISP backbone with hundreds of border routers connecting to customer and peer networks. Although we would ideally have iBGP sessions with all border routers, we could only collect data from the routers connecting to peer networks. Still, the BGP routing changes at these routers give us a unique view into the effects of BGP routing changes in the larger Internet on the ISP network. In addition, these border routers receive reachability information about customer prefixes via iBGP sessions with other routers, allowing us to analyze changes in how these border routers would direct traffic via customers. On a few occasions, our monitor experienced a temporary disruption in its iBGP connectivity to a border router; we preprocessed the BGP feeds as suggested in [23, 24] to remove the effects of these session resets.

The traffic data is collected from every border router by enabling Cisco’s Sampled Netflow [25] feature on all links. To reduce the processing overhead, flow records are sampled using techniques in [26]. Although sampling introduces inaccuracies in measuring small traffic volumes, this does not affect our system since we only use the traffic data to identify large traffic disruptions.

As shown in Table 2.2, our system significantly reduces the volume of data and produces only a few dozen large routing disruptions from millions of BGP updates per day from the periphery of the network. "Important" clusters in the table are clusters that affect more than 1% of total traffic volume in the network. In the remainder of the chapter, we present detailed results from the routing and traffic data collected continuously from August 16, 2004 to October 10, 2004—an eight-week period.

2.4 Tracking Routing Changes

In this section, we describe how we transform raw BGP update messages into routing events. We merge streams of updates from many border routers and identify changes from one stable route to another by grouping update messages that occur close in time. Along the way, we generate a report of prefixes that flap continuously.

2.4.1 Grouping BGP Updates into Events

A single network disruption, such as a link failure or policy change, can trigger multiple BGP messages as part of the convergence process [6, 7]. The intermediate routes are short-lived and somewhat arbitrary, since they depend on subtle timing details that drive how the routers explore alternate paths. To generate reports for the operators, we are interested in the change from one stable route to another rather than the details of the transition. As such, we group BGP updates for the same prefix that occur close together in time. Although previous studies, in particular BGP root-cause analysis, have followed a similar approach [10, 11, 12, 24, 27], we group the updates across *all* of the border routers since a single network disruption may cause multiple border routers to switch to new routes, and we wish to treat these as a single event.

We define an *event* as a sequence of BGP updates for the same prefix from any border router where the inter-arrival time is less than a predefined *event timeout*. Careful selection of the event-timeout value is important to avoid mistakenly combining unrelated routing changes or splitting a single change into two events. An appropriate event-timeout value can be determined by char-

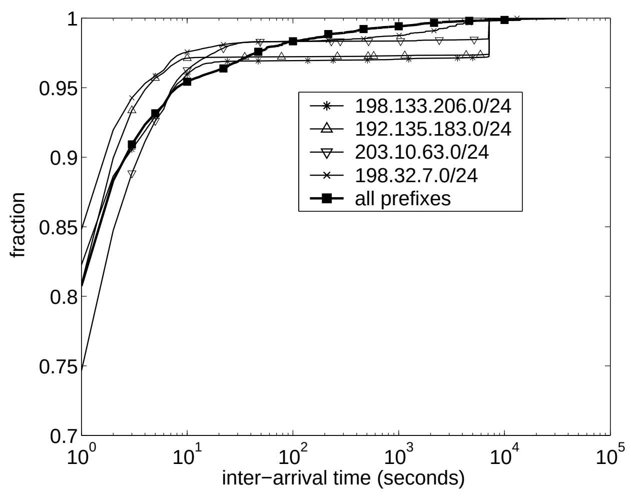

Figure 2.3: CDF of the BGP update inter-arrival time

acterizing the inter-arrival time of BGP updates in the network. For a controlled experiment, we analyze the inter-arrival times of BGP updates for public *beacon* prefixes that are advertised and withdrawn every two hours [28]; we also study the dynamics of the entire set of prefixes.

Figure 2.3 presents the cumulative distribution of the inter-arrival time of BGP updates for four beacons received from all of the border routers during a three-week period starting August 16, 2004, with the x-axis plotted on a logarithmic scale. More than 95% of the inter-arrival times are within a few tens of seconds; then the curves flatten until the inter-arrival time is around 7,000 seconds reflecting the two-hour advertisement period. In addition, previous studies have shown that the path-exploration process is often regulated by a 30-second *MinRouteAdvertisementInterval* (MRAI) timer [8]. As such, we choose an event timeout of 70 seconds, allowing the difference between the arrival times of updates at different vantage points to be as large as two MRAIs plus a small amount of variance. Looking across all prefixes in our dataset, about 98% of the updates arrive less than 70 seconds after the previous update.

2.4.2 Detecting Persistent Flapping

Certain prefixes never converge to a stable path due to persistent routing instabilities. Persistent flapping disrupts the reachability of the destination and imposes a significant BGP processing load on the routers, making it important for operators to detect and fix these problems. However, if we group updates for a flapping prefix using a 70-second timeout, the grouping process would continue

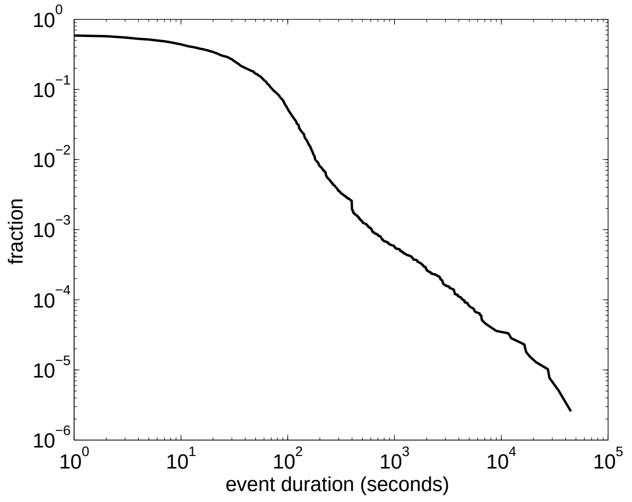

Figure 2.4: CCDF of event duration on a log/log scale

indefinitely. Instead, we generate a report once a sequence of updates exceeds a *maximum* duration, defined as the *convergence timeout*.

The convergence-timeout value should be large enough to account for reasonable convergence delays and yet small enough to report persistent flapping to the operators in a timely fashion. To identify an appropriate value, Figure 2.4 plots the complementary cumulative distribution function (CCDF) of event duration for the BGP updates in our network, with both axes on a logarithmic scale. More than 99% of events last less than a few hundred seconds, consistent with the findings in [6] that BGP typically takes less than three minutes to converge. As such, we select a convergence-timeout value of 600 seconds (10 minutes) for reporting flapping prefixes.

By applying our RouteTracker module to eight weeks of measurement data, we generated reports for about 23 prefixes per day, on average, though the number was as low as 7 on one day and as high as 46 on others. These persistently flapping prefixes were responsible for 15.2% of the total number of BGP update messages over the two-month period, though the proportion varied significantly from day to day (from 3.2% to 44.7%). These results were especially surprising given that all of the border routers were running route-flap damping [29], which is meant to suppress repeated updates of the same prefix. We identified three main causes of persistent flapping:

Unstable interface/session: Using syslog data [30] from the border routers, we determined that 3% of these updates (0.456% of the total number of updates) were caused by repeated failures of a flaky edge link or eBGP session. The prefixes were advertised each time the link/session

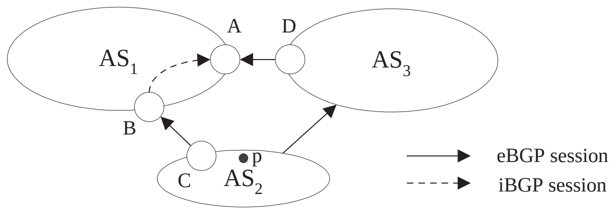

Figure 2.5: Persistent flapping due to failure of link B–C

came online, and withdrawn when the link/session failed. In Figure 2.5, the routers in AS_1 prefer the BGP route advertised by the customer AS_2 over the BGP route advertised by the peer AS_3. However, a flaky link between routers B and C would lead the routers in AS_1 to repeatedly switch between the stable route via AS_3 and the unstable route via AS_2. Route-flap damping did not stop AS_1 from using the unstable route from AS_2 for two reasons: (i) today's routers reinitialize the damping statistics associated with an eBGP session after a session reset and (ii) routers do not perform route-flap damping on iBGP sessions. In the short term, operators could respond to these cases by disabling (and ultimately repairing) the flaky link or session; in the longer term, router vendors could change the implementation of route-flap damping to prevent the persistent flapping.

MED oscillation: Through closer inspection of the BGP update messages and discussions with the operators, we determined that 18.3% of these updates (2.78% of the total) were caused by protocol oscillation due to the Multiple Exit Discriminator attribute. Unlike the other steps in the decision process in Table 2.1, the MED comparison is applied only to routes with the same next-hop AS. As a result, the BGP decision process does *not* impose an ordering on the routes in the system: a router may prefer route a over route b, b over c, and c over a. In the absence of an ordering of the routes, the routers may switch continuously between routes [31, 32]. Upon detecting a MED oscillation problem, the operators can request that the neighboring AS use a different mechanism to express its preferences for where it wants to receive the traffic destined for these prefixes (*e.g.*, RFC 1998 [33]).

Conservative flap-damping parameters: The remaining 78.6% of these updates (11.9% of the total) correspond to repeated advertisements and withdrawals by a neighboring AS. By inspecting the configuration of the routers, we verified that the flap-damping parameters assigned for these prefixes were not sufficient to dampen the instability. Using different parameters for different prefixes is not uncommon and is, in fact, recommended [34]. For example, ASes are advised to more heavily penalize the (many) smaller address blocks and to disable damping on critical prefixes (*e.g.*, the subnets that contain the Internet's root DNS servers). Upon noticing persistent flapping that is evading the damping algorithm, the operator could contact the neighboring AS to investigate the root cause or tune the router configuration to apply more aggressive damping parameters.

2.5 Classifying Routing Changes

In this section, we describe how we classify events to generate useful reports for the operators and to facilitate the clustering of related events in the next section. Since the current measurement infrastructure collects the BGP data only from the border routers connecting to peer networks, the following analysis is applied to the prefixes learned exclusively from peer ASes.

2.5.1 Merging Routes from Border Routers

To handle the large volume of BGP data arriving from the many border routers, EventClassifier needs a succinct representation of the routing state as it evolves over time. Rather than considering every BGP attribute, we focus our attention on how traffic entering at a border router would leave the AS en route to the destination prefix p. A border router BR_j may select a route R_p^j learned directly from one of its eBGP neighbors; in this case, we say that BR_j has route R_p^j with the next-hop address $nhop_p^j$ corresponding to the eBGP neighbor and a $flag_p^j$ of $\mathbf{e}$ for external. Alternatively, a border router BR_j may select as R_p^j a route learned via iBGP from another border router, resulting in a next-hop address $nhop_p^j$ of the remote border router and a $flag_p^j$ of $\mathbf{i}$ for internal. In a network with n border routers $BR_1, BR_2, \ldots, BR_n$, we have a route vector (r-vector) for prefix p of

$$RV_p = \langle R_p^1, R_p^2, ..., R_p^n \rangle$$

where the jth element $R_p^j = (nhop_p^j, flag_p^j)$ represents the *best* route for prefix p at router BR_j. By analyzing the evolution of RV_p, we can identify and classify the routing changes that affect how traffic leaves the AS, while ignoring changes in other BGP attributes (*e.g.,* downstream AS path or BGP community) that are beyond the operators' control.

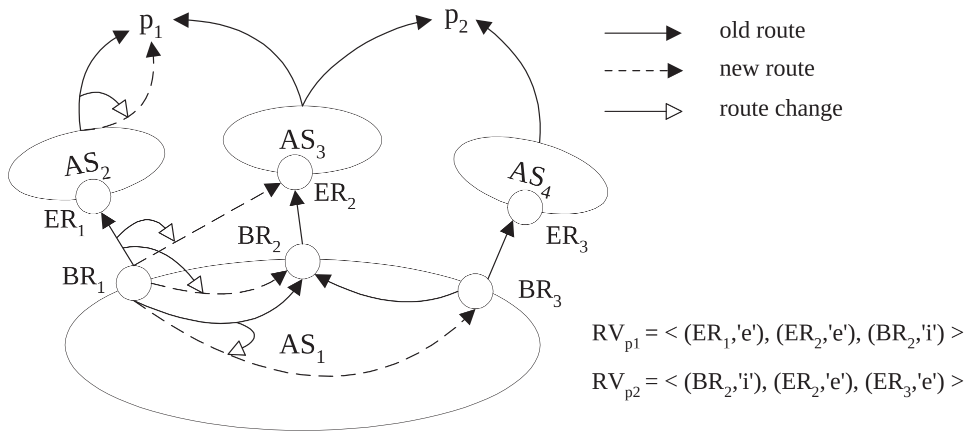
p1
p2
old route
new route
route change
AS2
AS3
AS4
AS1
ER1
ER2
ER3
BR1
BR2
BR3
RVp1 = < (ER1,'e'), (ER2,'e'), (BR2,'i') >
RVp2 = < (BR2,'i'), (ER2,'e'), (ER3,'e') >

Figure 2.6: R-vector element changes

2.5.2 Classifying Routing Events

When the network changes from one set of stable routes to another, comparing the old and new r-vectors (RV_p^{old} and RV_p^{new}, respectively) sheds light on the reason for the change and the effects on the traffic. We first describe the types of changes that each border router might experience and then present five event categories that consider the behavior across all of the routers.

A. Types of Events at One Border Router

To illustrate the types of routing events, Figure 2.6 shows examples for two destination prefixes. For prefix p_1, border routers BR_1 and BR_2 have eBGP-learned routes through AS_2 and AS_3, respectively; border router BR_3 selects an iBGP-learned route through BR_2. For prefix p_2, border routers BR_2 and BR_3 have eBGP-learned routes through AS_3 and AS_4, respectively; border router BR_1 selects an iBGP-learned route through BR_2. The dashed lines represent different ways an event can affect BR_1's routing decision, as summarized in Table 2.3:

No change: The border router BR_j may undergo a transient routing change only to return to the same stable best route. More generally, the BGP route may change in some attribute that is not captured in R_p^j. In Figure 2.6, a change in how AS_2 reaches p_1 does not necessarily change BR_1's decision to direct traffic via AS_2. For all of these scenarios, traffic entering the network at router j destined for the prefix p would continue to flow through the AS in the same way.

Internal path change: An internal event may cause a router to switch from one egress point

Type of Change for R_p^j	**Definition**
No change	$flag_p^{j,old} = flag_p^{j,new}$ $nhop_p^{j,old} = nhop_p^{j,new}$
Internal path change	$flag_p^{j,old} = flag_p^{j,new} = \mathbf{i}$, $nhop_p^{j,old} \neq nhop_p^{j,new}$
Loss of egress point	$flag_p^{j,old} = \mathbf{e}$, $flag_p^{j,new} = \mathbf{i}$
Gain of egress point	$flag_p^{j,old} = \mathbf{i}$, $flag_p^{j,new} = \mathbf{e}$
External path change	$flag_p^{j,old} = flag_p^{j,new} = \mathbf{e}$, $nhop_p^{j,old} \neq nhop_p^{j,new}$

Table 2.3: The types of change for r-vector element R_p^j

to another. In this case, router j uses an iBGP-learned route before and after the routing change (*i.e.*, $flag_p^{j,new} = flag_p^{j,old} = \mathbf{i}$) but with a different next-hop router (*i.e.*, $nhop_p^{j,new} \neq nhop^{j,old}$). In Figure 2.6, a change in the IGP topology could make BR_1 see BR_3 as the *closest* egress point for reaching prefix p_2, instead of BR_2.

Loss of egress point: An external event may cause a route to disappear, or be replaced with a less attractive alternative, forcing a border router to select an iBGP route. In this case, a router BR_j has $flag_p^{j,old} = \mathbf{e}$ and $flag_p^{j,new} = \mathbf{i}$. In Figure 2.6, suppose AS_2 withdraws its route for p_1 and that BR_1 has no other eBGP-learned routes; then, BR_1 would select the iBGP-learned route from BR_2. This routing change would force the traffic that used to leave the network at BR_1 to shift to BR_2.

Gain of egress point: An external event may cause an eBGP-learned route to appear, or be replaced with an attractive alternative, leading a border router to switch from an iBGP-learned route to an eBGP-learned one. In this case, a router BR_j has $flag_p^{i,old} = \mathbf{i}$ and $flag_p^{j,new} = \mathbf{e}$. In Figure 2.6, suppose AS_2 starts advertising a route to p_1 again; then, BR_1 would start using the eBGP-learned route, causing a shift back to BR_1.

External path change: An external event may cause a router to switch between eBGP-learned routes with different next-hop ASes. In this case, the $flag_p^j$ remains at $\mathbf{e}$ while the next hop changes (*i.e.*, $nhop_p^{j,new} \neq nhop_p^{j,old}$). In Figure 2.6, suppose AS_2 withdraws the route for p_1, causing BR_1 to switch to an eBGP-learned route from AS_3. Then, BR_1 would start directing traffic to a different egress link at the same router.

Event Category	Events	Updates	Upd./Ev.
Distant/transient disruption	50.3%	48.6%	12.6
Internal disruption	15.6%	3.4%	2.9
Single external disruption	20.7%	7.9%	5.0
Multiple external disruption	7.4%	18.2%	32.0
Loss/gain of reachability	6.0%	21.9%	47.9

Table 2.4: Event distribution in updates

B. Classes of Route-Vector Changes

Since each of the n elements in the r-vector can have five different types of changes, routing events could fall into 5^n different categories, which would be extremely unwieldy for generating reports for network operators. Instead, we classify the events based on the *severity* of the impact on the traffic, leading to five disjoint categories:

Distant/transient disruption: Some events do not have any influence on the flow of traffic through the AS. We define an event as *distant or transient disruption if each element of the r-vector has "no change."* A distant routing change that occurs more than one AS hop away does not affect the R_p^j values. A transient disruption may cause temporary routing changes before the border routers converge back to the original BGP routes. These events are worthwhile to report because the downstream routing change may affect the end-to-end performance (*e.g.,* by changing the round-trip time for TCP connections) and the convergence process may lead to transient performance problems that can be traced to the routing event. As shown in Table 2.4, this category explains about half of the events and half of the BGP update messages; these events trigger an average of 12 or 13 update messages for the BGP convergence process.

Internal disruption: An internal event can cause a router to switch from one internally-learned route to another. We define an event as an *internal disruption if the change of each of the elements in its r-vector is either of type "no change" or of type "internal path change", with at least one element undergoing an "internal path change."* Caused by a change in the IGP topology or an iBGP session failure, these events are important because they may cause a large shift in traffic as routers switch from one egress point to another [22, 35]. As shown in Table 2.4, internal disruptions account for about 15% of the events and just 3.4% of the updates; on average, an internal event triggers just a few iBGP update messages as some routers switch from one existing route to another.

Single external disruption: Some events affect the routing decision at a single border router for an eBGP-learned route. We define an event as a *single external disruption if only one r-vector element has a change of type "loss of egress point," "gain of egress point," or "external path change."* Typically, an ISP has eBGP sessions with a neighboring AS at multiple geographic locations, making it interesting to highlight routing changes that affect just one of these peering points. These kinds of events cause a shift in traffic because routers are forced to select an egress point that is further away [36]. For example, a single external disruption may arise because an eBGP session between the two ASes fails, forcing the border router to switch to a less-attractive route. As shown in Table 2.4, these disruptions account for over 20% of the events and nearly 8% of the updates; since these localized events affect a single router, the number of update messages per event is limited.

Multiple external disruptions: In contrast to the previous category, some events affect more than one border router. We define an event as a *multiple external disruption if multiple r-vector elements have a change of type "loss of egress point," "gain of egress point," or "external path change," and the r-vector includes at least one eBGP-learned route before and after the event.*[3] In Figure 2.6, if the owners of prefix p_1 changed providers to start using AS_4 instead of AS_2 and AS_3, every border routers in AS_1 would experience a disruption. As shown in Table 2.4, this category accounts for just over 7% of events and 18% of updates; the large number of update messages stems from the convergence process where multiple border routers must explore alternate routes.

Loss/gain of reachability: An event may cause a prefix to disappear, or become newly available. We define an event as *loss of reachability if every r-vector element with an external route experiences a "loss of egress point."* A loss of reachability is extremely important because it may signify a complete loss of connectivity to the destination addresses, especially if the routers have no route for other prefixes (*e.g.,* supernets) covering the addresses. Similarly, we define an event as *gain of reachability if initially no eBGP-learned routes exist and at least one r-vector element experiences a "gain of egress point."* In some cases, the *gain* of reachability is indicative of a problem, if the network does not normally have routes for that prefix. For example, a neighboring AS may mistakenly start advertising a large number of small subnets; overloading the memory resources on the router may have dire consequences, such as crashing the network [37]. As shown in Table 2.4, this category accounts for 6% of the events and nearly 22% of the update messages;

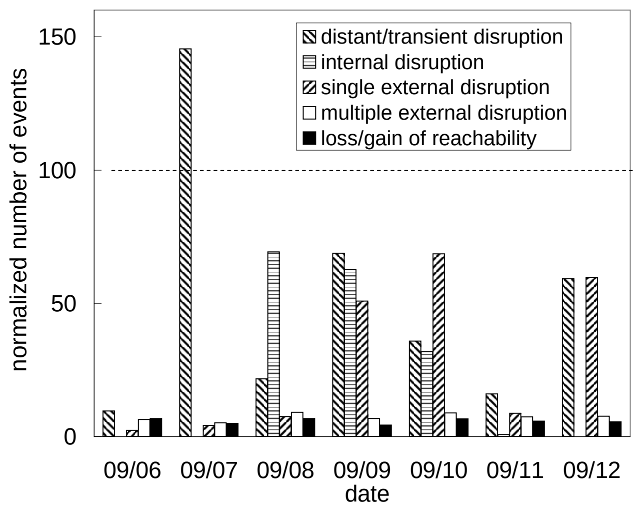

Figure 2.7: The (normalized) # of daily events by category.

the gain or loss of reachability often triggers a large number of update messages as every border router explores the many alternate routes.

Overall, the severity of the external events increases from single external disruptions to multiple external disruption, and ultimately to loss/gain of reachability. In general, the number of events in the "loss/gain of reachability" and "multiple external disruption" is stable over time, whereas the other categories vary significantly. Figure 2.7 shows the number of daily events (where 100 represents the average number of events per day over the eight-week study) for each event category during the week of September 6-12, 2004. For example, September 7 had a large number of distant/transient disruptions, and some days see a much larger number of internal disruptions and single external disruptions than others. The high variability arises from the fact that network disruptions can occur at arbitrary times and may affect a large number of destination prefixes, as discussed in the next section. Given the high variability in the number and type of events, predicting them in advance and overprovisioning for them is very difficult, making it even more important for operators to learn about disruptions as they occur to adapt the configuration of the network.

2.6 Grouping Related Events

In this section, we describe how to identify related events across *time* and *prefixes*. By clustering events across time for the same prefix, we identify destination prefixes that have unstable routes. By clustering events of the same type across prefixes, we group events that appear to have a common cause. We present techniques to identify groups of prefixes affected by hot-potato routing changes and eBGP session resets, which are responsible for many of the large clusters. We validate our inferences using RouteViews data [38], syslog reports [30], and an independent analysis [35] of internal topology changes.

2.6.1 Frequently Flapping Prefixes

Some destination prefixes undergo frequent routing changes that introduce a large number of events in a relatively short period of time. In contrast to the persistent flapping analyzed in Section 2.4.2, these routing changes occur at a low enough rate to span multiple events. For example, a prefix may have a long-term instability due to flaky equipment that fails every few minutes, falling outside of our 70-second window for grouping BGP updates into events. Even if the equipment fails at a higher rate, the BGP updates may be suppressed periodically due to route-flap damping [29], leading to multiple events. Identifying these slowly *frequently flapping* prefixes is important for addressing long-term reachability problems and for reducing the number of BGP updates the routers need to handle.

To identify frequently flapping prefixes, we group events for the same destination prefix that occur *close together in time* (with an inter-arrival time less than $thresh_T$), and flag cases where the *number of events exceeds a predefined threshold* (max_count). We implement this heuristic by keeping track of each prefix that has had an event in the last $thresh_T$ seconds, along with the time of the last event and a count of the total number of events. Upon learning about a new event from RouteTracker, we check if the prefix has experienced an event in the last $thresh_T$ seconds and update the timestamp and counter values; once the counter exceeds max_count, we generate a report.

Since route changes can happen on virtually any timescale, the parameters $thresh_T$ and max_count should be set to highlight the most unstable prefixes without generating an excessive number of reports. Figure 2.8 shows the complementary cumulative distribution of the number of events per cluster over our eight-week measurement period. For all three values of $thresh_T$, more than 99% of the clusters have fewer than ten events; still, a small number of very large clusters exist. Having a very small $thresh_T$ might cause our system to overlook some unstable prefixes with a long

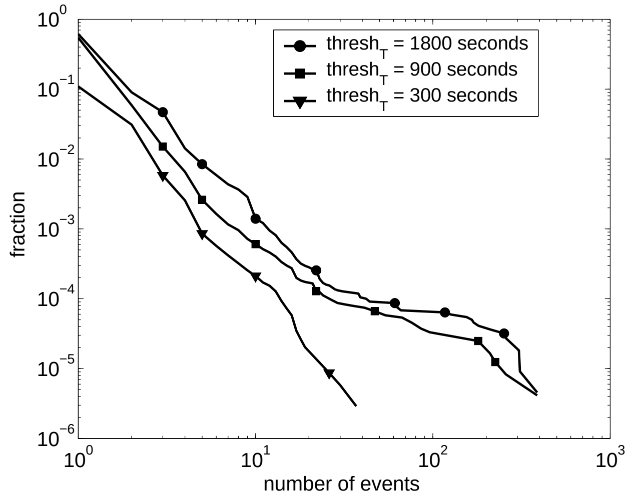

Figure 2.8: CCDF of the number of events per cluster for event correlation across time

cycle between routing changes. For example, a prefix that has a routing change every ten minutes would not be detected by a $thresh_T$ of 300 seconds. Based on the results in Figure 2.8, we assign $thresh_T$ to 900 seconds and max_count to 10 to draw attention to the small number of very unstable prefixes.

In our analysis, the percentage of events caused by frequently flapping prefixes varies from day to day from a low of 0.41% to a high of 32.78%, with an average of 3.38%. Most of these events are in category "loss/gain of reachability." We believe that frequent flapping tends to originate near the destination, making these instabilities visible to other ASes. To validate our inferences, we applied our heuristic for identifying frequently flapping prefixes to the BGP data from RouteViews [38]. For the week of September 26 to October 2, 2004, all 35 prefixes we identified were also flapping frequently in at least one other vantage point in the RouteViews data. Whether (and how) operators react to frequently flapping prefixes depends on the network responsible for the problem. If the frequent flapping comes from one of the ISP's own customers, the operators may be able to work with the customer to identify and fix the problem. If the flapping comes directly from a peer network (or one of the peer's customers), the operators may contact the peer to request that the peer address the problem.

2.6.2 Disruptions Affecting Multiple Prefixes

A single disruption (such as a link failure or a policy change) may affect multiple prefixes in a similar way, in a very short period of time. Grouping these prefixes together magnifies the visibility of the common effects and substantially reduces the number of reports for the operators. The five categories identified in Section 2.5.2 provide an effective way to identify prefixes affected in a "similar way." In addition, we also consider whether the border routers changed from a better route to a worse route, a worse route to a better route, or between two equally-good routes, in terms of the first six steps of the decision process in Table 2.1. This distinction gives us insight into whether the old route was withdrawn (or replaced by a less-attractive route), the new route recently appeared (or was replaced by a more-attractive route), or the router switched between two comparable routes (*e.g.*, because of a change in the IGP path costs).

In particular, we group events for different destination prefixes that (i) belong to the same category (using the taxonomy from Section 2.5.2), (ii) undergo the same kind of transition (from better to worse, or worse to better), and (iii) start no more than $thresh_P$ seconds after the first event. We consider the start time of the events because the first update is most likely to be directly triggered by the network event. We implement this heuristic by keeping track of the identifying information for each cluster (*i.e.*, the event category and the kind of transition) as well as the time of the first event and a count of the number of events. Upon generating a new event, we check if the event matches with the identifying information and arrives within $thresh_P$ seconds after the first event in the cluster. The correlation process adopts a clustering algorithm similar to those used in previous BGP root-cause analysis studies [10, 11, 12].

Setting $thresh_P$ too small runs the risk of splitting related events into two clusters. If a network disruption affects a large number of prefixes, the effects could easily spread over several tens of seconds. For example, a BGP session failure or hot-potato routing disruption that affects tens of thousands of prefixes requires the router to send numerous update messages, which could easily take up to a minute [35]. To account for these effects, we carefully select a value of 60 seconds for $thresh_P$ after a study of the duration traditional routing changes (*e.g.*, session resets) normally take to affect all of their related prefixes. Since $thresh_P$ is used to compare the start times of the two events, our heuristic cannot assume that a cluster is complete once the current time (the time of newly arrived BGP update in the system) is $thresh_P$ after the time of the first

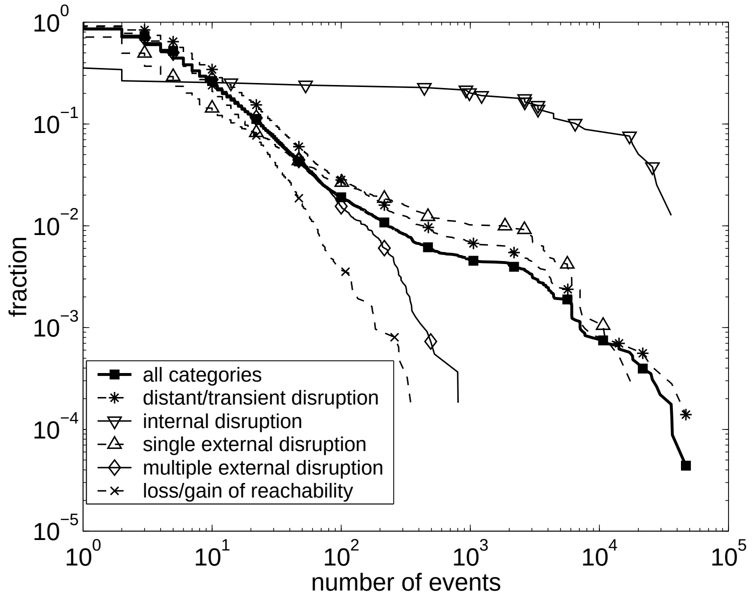

Figure 2.9: CCDF of the number of event per cluster for event correlation across prefixes

event in the cluster since an event may still be "in progress." Knowing that an event lasts at most the convergence timeout (from Section 2.4.2), in our heuristic, each cluster waits for a total of $thresh_P + convergence\_timeout$ to ensure that no ongoing, correlated events should be included in the cluster. In total, then, our heuristic waits for 660 seconds before declaring a cluster complete.[4]

Figure 2.9 shows the effectiveness of clustering in combining related events. The graph plots the complementary cumulative distribution of the number of events per cluster over the eight-week period, on a log-log scale. Although 99% of the clusters have less than a hundred events (as shown in the "all categories" curve), a few clusters have a tremendous number of events. Meanwhile, the curves for different categories of events have distinctive characteristics. The categories "multiple external instability" and "loss/gain of reachability" have much smaller clusters, while the other three categories have some very large clusters with tens of thousands of affected prefixes. The categories "internal disruption" and "single external disruption" tend to have larger clusters than the other categories. Next, we show that these very large clusters stem from hot-potato routing changes and eBGP session resets, respectively.

A. Hot-Potato Changes

According to the BGP decision process in Table 2.1, a router selects among multiple equally good BGP routes (*i.e.*, routes that have the same local preference, AS path length, origin type, MED value, and eBGP vs. iBGP learned) the one with the smallest IGP cost. Such routing practice is called *hot-potato* routing [35]. An IGP topology change can trigger routers in a network to select a different *equally good* BGP route for the same prefix, and these changes may affect multiple prefixes. This section describes the routing disruptions caused by these hot-potato changes.

"Hot-potato" changes only affects the egress points each router selects for the prefixes. As the event classification in Section 2.5.2, it results in "internal disruptions" to the network. After the correlation process, the event cluster in category "internal disruption" magnifies the impact of the "hot-potato" changes. When these kinds of disruptions occur, the operators need to know which routers and prefixes are affected to gauge the significance of the event. Such information can be obtained by comparing the old and new r-vectors for all of the events in the cluster because each element in the r-vector carries the next-hop address for the corresponding router.

A previous study [35] proposed a heuristic for identifying hot-potato routing changes at a single router, based on a single stream of BGP updates from that router and data from an IGP topology monitor. Applying this technique to specific ingress routers allowed us to make direct comparisons between the two approaches. For the period from August 16 to September 30, 2004, over 95% of the large clusters (*i.e.,* clusters with more than 1000 events) of internal disruptions identified by our system are also identified using the technique in [35]. Inspecting the other 5% of cases in more detail, we discovered that these clusters corresponded to the restoration of a link in the network, where the failure had caused a previous hot-potato routing change that was detected using both techniques. As such, we believe that these disruptions are hot-potato routing changes that were not detected by the heuristic in [35].

B. eBGP Session Resets

The failure or recovery of an eBGP session can cause multiple events that affect the eBGP-learned routes from one neighbor at a single border router. Upon losing eBGP connectivity to a neighbor, a border router must stop using the routes previously learned from that neighbor and switch to less-attractive routes. The border router may switch to an eBGP-learned route from a different neighbor, if such a route exists; this would result in an “external path change” for the destination prefix. Alternatively, the router may have to switch to an iBGP-learned route from a different border router; this would result in a “loss of egress point” for the destination prefix. When the session recovers, the border router learns the BGP routes from the neighbor and switches back to the eBGP-learned routes advertised by this neighbor for one or more destination prefixes (causing either an “external path change” or a “gain of egress point”).

To identify a session failure, we first group events that (i) belong to the category “single external disruption,” (ii) have an *old* route with the same border router and neighbor (*i.e.,* the same $R_p^{j,old}$), (iii) have a routing change that goes from better to worse, and (iv) occur close together in time. However, this is not enough to ensure that the session failed, unless the router has stopped using most (if not all) of the routes previously learned from that neighbor. As such, we also check that the number of prefixes using the neighbor has decreased dramatically.[5] Similarly, to identify a session recovery, we first group events that (i) belong to the category “single external disruption,” (ii) have a *new* route with the same border router and neighbor (*i.e.,* the same $R_p^{j,old}$), (iii) have a routing change that goes from worse to better, and (iv) occur close together in time, and also involve a significant increase in the number of prefixes associated with that neighbor, back to the expected level.

Applying our heuristic to the “single external disruption” clusters that contain more than 1000 events, we found that 95.7% of these large clusters were linked to an eBGP session going up or down. To validate our inferences, we consulted the syslog data [30], which reports when the status of a BGP session changes. The syslog data confirmed more than 95% of our inferences. Our inferences not only captured all of the resets in syslog but identified a few disruptions that were not reported by syslog. Interestingly, we sometimes found that our analysis suggests that the session failure occurred up to ten seconds *before* the entry in the syslog data. After checking for possible timing discrepancies between the BGP and syslog data, we speculate that the remote AS is shutting down the BGP session in a graceful manner by first *withdrawing* all of the routes before actually disabling the session. This practice highlights the importance of using an algorithm such as ours even when syslog data are available.[6] A complete loss of the routes from a neighbor does *not* necessarily arise only from a session failure. Instead, the neighbor’s router may be reconfigured with a new policy (*e.g.*, that withdraws the previous routes) or lose connectivity to other routers in its own network. These kinds of disruptions could have a significant impact on traffic inside an AS, and would not generate a syslog report. The influence of large disruptions on the traffic is explored in more detail in the next section.

2.7 Estimating Traffic Impact

We now describe the final component of the system—TrafficMeter which allows us to estimate the traffic impact of the routing disruptions produced by the EventCorrelator. Although the traffic volume on a link typically varies gradually across days and weeks, sudden changes in traffic can lead to congestion in some parts of the network. A recent study [39] shows BGP routing disruptions are responsible for many of the largest traffic shifts in backbone networks. Below we first discuss how we compute traffic weights to estimate the impact on traffic and then focus on two types of routing disruptions with the most impact.

2.7.1 Computing Traffic Weights

TrafficMeter aggregates the Netflow data [25] collected on the outgoing links to compute prefix-level traffic statistics. For each destination prefix, we define a *traffic weight* that corresponds to the percentage of traffic destined to that prefix across the overall traffic volume in the network. In essence, the weight corresponds to the relative popularity of the prefix. Since the proportion of traffic destined to each prefix changes over time, we compute the weights over a sliding time window (*e.g.*, the last month). The weights allow us to estimate the potential impact of a cluster of routing events by considering the sum of the weights for all prefixes in the cluster. Although the weights do not capture the variations in traffic per prefix across time and location, they do provide a simple way to flag routing disruptions that affect clusters of prefixes that attract a high volume of traffic.

In Figure 2.10, we plot the complementary cumulative distribution of traffic weight of a prefix, an event, and an event cluster over the eight-week period of our study. The "prefix" curve shows the significant differences in popularity of the prefixes, consistent with previous studies [24, 40]. Interestingly, the "event in all categories" curve looks largely the same, suggesting that routing events

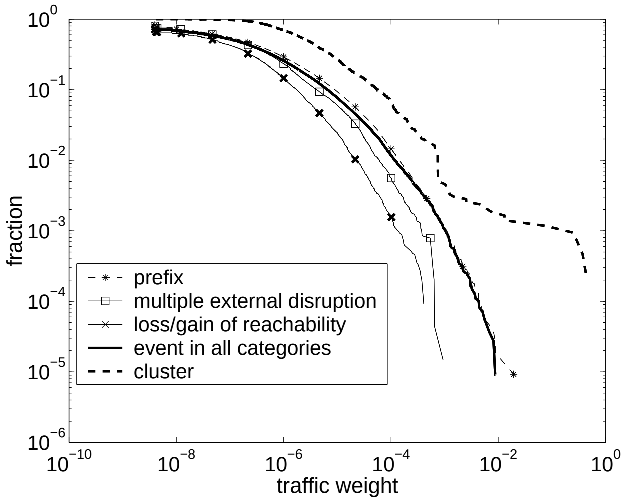

Figure 2.10: CCDF of traffic weight

affect prefixes across the entire range of popularities. This occurs because the many events in categories “distant/transient disruption,” “single external disruption,” and “internal disruption” tend to affect a wide range of destination prefixes, largely independent of their popularity; the curves for these three categories of events are not shown, as they look almost identical to the “prefix” and “event in all categories” curves. In contrast, the curves for events in categories “multiple external disruption” and “loss/gain of reachability” suggest that these events tend to involve prefixes that receive less traffic.

The “cluster” curve plots the distribution of traffic weight across the event clusters. As expected, a cluster tends to have a large traffic weight since it combines one or more related events. The tail of the curve suggests that a small number of clusters are responsible for a significant portion of the large traffic shifts. Meanwhile, our results reveal that these “significant” clusters have a large number of events, implying the routing change affects many prefixes. Our system observes a few dozen such large clusters each day and highlights them for the network operators for their attention. We use the threshold of 1% for traffic weight to signal significant routing disruptions, since the vast majority of clusters fall below that threshold. This avoids operators focusing their attention on the many BGP disruptions that affect a very small fraction of the traffic.

2.7.2 Disruptions With Large Weights

We now discuss our empirical findings using TrafficMeter based on our eight weeks of measurement data. Interestingly, most big events in terms of the amount of traffic weight are single external disruptions and internal disruptions. Thus, we focus on those in Figure 2.11 showing the duration of a routing disruption relative to the corresponding traffic weight of the affected prefixes for clusters with traffic weight larger than 1%. On average, internal disruptions (*e.g.,* hot potato changes) result in larger traffic weights than single external disruption (*e.g.,* session resets), because internal routing disruptions usually affect multiple locations. They also appear to have longer durations than single external disruptions. Long-lived events allow operators to adapt routing configurations as needed to alleviate possible network congestion. Our tool highlights only a few critical events which are both long-lived and expected to affect a large amount of traffic. This helps focus operators' attention on routing disruptions where mitigation actions, such as tuning the routing protocol configuration, might be necessary.

Figure 2.11 also shows that our tool captures some large disruptions that are short-lived, lasting 30 seconds to a few minutes. In addition to most of the "single external disruption" points in the graph, these short-lived disruptions include many large clusters in the "distant/transient disruption"; this category accounts for 78.8% of all event clusters with traffic weight higher than 1%. These clusters involve events that start and end with the same route vectors, with some sort of transient disruption in between. Although short-lived traffic shifts do not have a sustained impact on network load, users may encounter brief periods of degraded performance that could be traced to these disruptions. Interestingly, these short-lived traffic shifts are extremely difficult to detect using conventional measurement techniques, such as SNMP and Netflow, that aggregate traffic statistics on the timescale of minutes. In contrast, our troubleshooting system can identify short-lived routing disruptions that may have large effects on user performance.

2.8 System Evaluation

In this section, we demonstrate that our system imposes a small amount of memory and CPU processing overhead to run in real time on a commodity computing platform. Throughout the evaluation of our system on eight weeks of data, the system memory footprint never exceeded 900

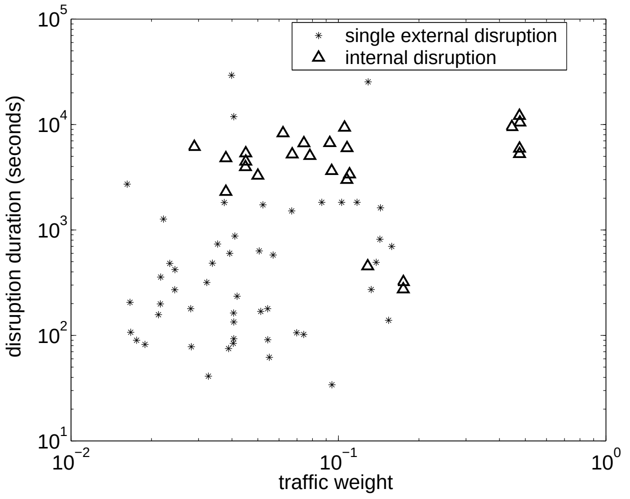

Figure 2.11: Routing disruption durations vs. traffic weights

Megabytes and every interval of 70 seconds of BGP updates was processed in less than 70 seconds.

We characterize the system performance through an off-line emulation over the past measurement data. Due to operational concerns, our system could not access the collected data in real-time. Instead, we stored the measurements locally and replayed the data in our tool. We ran our tool on a Sun Fire 15000 equipped with several 900 MHz Ultrasparc-II processors. Only one processor was used during the experiments. We evaluate the system using two metrics: *memory usage* and *execution speed*.

2.8.1 Memory Usage

The memory usage in our troubleshooting system consists of two parts: *static* usage and *dynamic* usage. The static memory is allocated to store the best route for each border router and destination prefix. In the core of today's Internet, each router learns reachability information for about 160,000 prefixes (also confirmed by RouteViews [38]). The total static memory usage in our system is about 600 Megabytes.

Dynamic memory, on the other hand, is allocated to maintain the data structures continuously created in response to the arrival of BGP updates. The essential data objects kept in the system are clusters, whose memory are dynamically allocated and reclaimed during the process as discussed in Section 2.6. In processing the eight weeks of measurement data, the dynamic memory footprint

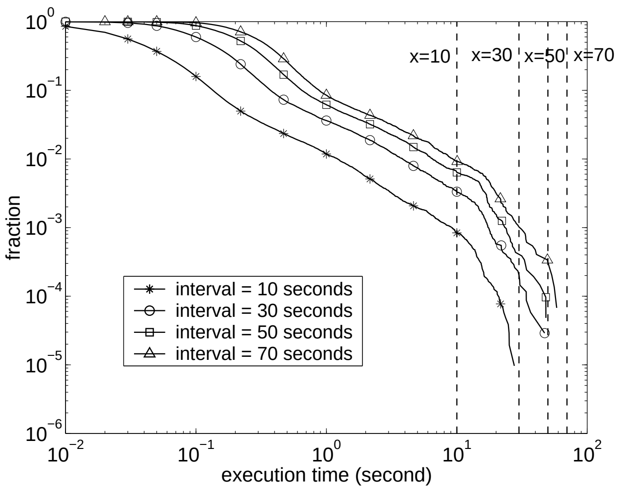

Figure 2.12: System execution speed

of the system never exceeded 300 Megabytes.

2.8.2 Execution Speed

We measure how quickly the system processes the BGP updates. Because the progression of each BGP update in the system varies depending on the expiration condition of several timers, we have conducted the experiment for each BGP update sequence within a fixed time interval called *epoch*, rather than characterizing the execution latency of each individual BGP update. During each test, we randomly selected a starting point in the eight-week BGP update sequence and then divided the subsequent BGP update stream into non-overlapping epochs. Then we measured the execution time for each epoch of a fixed epoch interval. We varied the epoch interval among the values of 10, 30, 50, 70 seconds. Because the machine is a time-sharing system, we ran each experiment three times to ensure the accuracy of the measurement results; we saw virtually no variation in the results across the three experiments.

Figure 2.12 shows the complementary cumulative distribution of the execution time for each of the four epoch intervals. As shown in the graph, the execution of nearly every epoch was completed within the epoch interval. For example, the curve for a ten-second epoch interval shows that more than 99% of epochs could be processed within one second; however 0.1% of the epochs required more than ten seconds to complete. Our system occasionally lags behind the arrival of BGP updates, due to the bursty arrival pattern of BGP updates. Our data show that, while the average number of BGP updates per second is well below 100 (which corresponds to about 30 Kbps data rate), the maximum number of BGP updates received in our system in one second could well exceed 10,000 (which corresponds to 3 Mbps data rate).

Despite the existence of execution lags, for an epoch interval of 30 seconds, its percentage becomes much smaller (0.01%) by smoothing the BGP update bursts with a longer interval. The execution lag is completely eliminated when we set the epoch interval to 70 seconds; that is, every interval of 70 seconds worth of BGP updates was completely processed in less than 70 seconds. We believe the occasional execution lag is acceptable. Recall that each event is identified only if at least a period of event timeout elapses after the arrival of the last BGP update in the event. Typically the timeout value is a few tens of seconds (70 seconds, in our experiments). That is, even with instantaneous processing, each BGP update would have to wait for at least 70 seconds before a report is generated for the network operators. As such, smoothing the processing of BGP updates over a few tens of seconds does not introduce a problem.

2.9 Related Work

There is a large body of literature on characterizing BGP data using passive monitoring [6, 7, 23, 41, 24] as well as active route injection [28]. Our study is also preceded by several recent efforts [10, 11, 12, 13, 14] to identify the location and cause of routing changes by analyzing BGP update messages along three dimensions: time, views, and prefixes. Our work is similar in that we analyze BGP data along the same dimensions to group related routing changes. However, we focus on organizing large volumes of BGP updates seen in a single AS in real time into a small number of reports belonging to categories directly useful to operators to help mitigate the problems.

In analyzing BGP data collected from multiple vantage points within a single AS, our work is similar to the BorderGuard [36] study that identifies inconsistent routing advertisements from peers. In contrast, we classify *all* routing changes seen by the border routers into useful categories. The work in [22] presents a strawman proposal where each AS collects BGP data from its border routers as part of an end-to-end service for identifying the location and cause of routing changes. Each AS uses the data to detect and explain its own *internal* routing changes, rather than trying to detect and diagnose interdomain routing events. Recent work [42] has considered how to detect

Component	Types of Report	Information
RouteTracker	Persistently flapping prefixes	Prefix, time duration, AS paths
EventCorrelator	Frequently flapping prefixes	Prefix, time duration, AS paths
TrafficMeter	Transient disruption clusters	Time, traffic weight
	eBGP session reset clusters	Time, eBGP session, traffic weight
	Hot-potato change clusters	Time, prefix matrix, traffic weight
	Other disruption clusters	Summary statistics

Table 2.5: Routing disruption reports

Component	Parameters
RouteTracker	Grouping: event timeout (70s)
	Flapping: convergence timeout (600s)
EventCorrelator	Time: $thresh_T$ (900s), max_count (10)
	Prefix: $thresh_P$ (60s)

Table 2.6: Summary of the system parameters

network anomalies through a joint analysis of traffic and routing data. This work looks for significant changes in both the volume of traffic and the number of update messages, without delving in to the details about the specific destination prefixes and event types involved.

2.10 Concluding Remarks

We have presented the design and evaluation of an online troubleshooting system for identifying important BGP routing changes in an IP network. Table 2.5 summarizes the types of disruption reports that are generated by our system to help operators improve the management of the network. In addition, as shown in Table 2.2, the reduction in the amount of raw information is significant—extracting a few dozen reports from millions of BGP updates collected from multiple vantage points at the periphery of the network. Using the concise r-vector data structure to capture BGP routing changes, we identified five categories of BGP routing disruptions that vary in the severity of the impact on the traffic. Table 2.6 summarizes the parameters and their recommended values used in our tool. They can also be adapted to reflect network conditions as well as operators' preferences. Applying the tool to eight weeks of routing and traffic data from a tier-1 ISP network, we identified several ways for operators to improve the routing stability of the network. Despite having route-flap damping features enabled on all of the routers, our tool surprisingly discovered a large number of updates from persistently flapping prefixes and identified three causes. Meanwhile, we found that hot-potato routing changes and eBGP session resets were responsible for many of the large routing disruptions.

CHAPTER 3

Internet Routing Resilience to Failures: Analysis and Implications

The network troubleshooting tool developed in Chapter 2 helps operators deploy countermeasures *in face of* failures. In the next two chapters, we undertake a different approach to enhance the robustness of the Internet interdomain routing. This chapter first proposes a measurement framework that systematically characterizes how the current Internet routing system reacts to various types of large-scale failures, and then pinpoint the reliability bottlenecks of the Internet.

3.1 Introduction

Given our growing dependence on the Internet for important and time-critical applications such as financial transactions and business operations, there is a strong need for high availability and good performance at all times for most network paths on the Internet. To provide such assurance, Internet routing plays a critical role, as its main function is to identify network paths with sufficient resources between any two network prefixes. However, it is well-known that interdomain routing on today's Internet is *policy-driven* to satisfy commercial agreements. Policy restrictions prevent the routing system from full exploitation of the underlying topology, as physical connectivity does not imply reachability. It is unknown how such restrictions affect the failure-resilience of the Internet routing system.

On average, routing on today's Internet works well, ensuring reachability for most networks and achieving reasonable performance over most paths. However, there is a serious lack of understanding of Internet routing resilience to significant but realistic failures such as those caused by the 911 event [19] and the Taiwan Earthquake in December 2006 [21]. For instance, for several ten minutes to hours after this earthquake many Asian sites of U.S. companies cannot communicate with their headquarters or data centers in North America, preventing important business operations. A particularly noteworthy observation is that due to the North-America-centric placement of most top-level DNS domain servers for *.COM* domain, some Asian Web users cannot reach even regional servers due to the inability to contact authoritative DNS servers.

In this chapter, we systematically analyze how the current Internet routing system reacts to various types of failures by establishing a realistic failure model, and then pinpoint reliability bottlenecks of the Internet. To achieve this, we first construct a topology graph which accurately captures the AS-level structure of today's Internet. Techniques are designed to address issues of topology completeness and relationship accuracy. Then we develop a generic failure model that captures the effect (not the cause) of most common failures affecting routing at the interdomain level. Note that such failures can also result from attacks instead of natural disaster. We attempt to identify critical links whose failures can cause large and severe impact on the Internet. They are effectively Achilles' heels of the Internet.

We develop a simulation tool to perform such what-if failure analysis to study routing resilience which is efficient to scale to Internet-size topologies. We focus on fundamental structural and policy properties that influence network resilience to failures. We attempt to draw conclusions independent of inaccuracies in relationship inference and topology construction by focusing on the underlying properties of networks that affect network-resilience properties. For example, there are a limited number of trans-oceanic links, which can easily become reliability bottlenecks. By focusing on the impact of structural and policy properties, our analysis provides guidelines for future Internet design.

The chapter is organized as follows. Section 3.2 introduces our overall methodology. The failure models used in our study as well as the corresponding empirical events are described in Section 3.3. The detailed resilience analysis under different types of failures using our simulation tool are discussed in Section 3.4. Finally, we discuss the related work and conclude the chapter.

3.2 Analysis Methodology

We describe our methodology for failure resilience analysis. It consists of three main components: (i) building the AS-level topology, (ii) inferring AS routing policies, and (iii) conducting failure analysis. Unlike previous studies, we carefully perturb relevant parameters.

3.2.1 Topology Construction

We use publicly available BGP data from a large number of vantage points in the form of routing table snapshots as well as routing updates to construct an *AS-level network topology*. Combining routing updates with tables improves the completeness of the topology by including potential backup paths revealed only during transient routing convergence. However, history data may also introduce inaccuracies in the network topology caused by AS links that are no longer valid. We would like to obtain a topology graph that is as complete as possible to avoid underestimating routing resilience of today's Internet. By including network paths obtained from history data that may no longer exist, we may nevertheless overestimate its failure resilience.

To balance between the completeness and accuracy of network topology, we use 2 months of routing data from RouteViews [38], RIPE [43], public route servers [44] as well as a large content distribution network from March to April, 2007. The measurement data were collected from vantage points located in a total of 483 different ASes. To reduce the size of the network graph and speed up our analysis, we prune the graph by eliminating *stub* AS nodes [45], defined to be customer ASes that do not provide transit service to any other AS. These can be easily identified from routing data as ASes that appear only as the last-hop ASes but never as intermediate ASes in the AS paths. As a result, we could eliminate 63% of the links and 83% of the nodes. For the analysis of routing resilience to failures, we can restore such information by tracking at each AS node in the remaining graph the number of stub customer nodes it connects to including information regarding whether they are single-homed or multi-homed to other ISPs.

3.2.2 Topology Completeness: Missing AS Links

The BGP data collected from a limited number of vantage points, such as RouteViews and RIPE, cannot locate all of the links in today's Internet [46, 47]. Certain links, especially peer-to-

Graph	# of nodes	# of links	# of peer-peer links	# of cust.-prov. links	# of sibling links
CAIDA	4342	14815	3558 (24.0%)	11168 (75.4%)	89 (0.1%)
SARK	4430	25485	3801 (14.9%)	21684 (85.1%)	0 (0.0%)
Gao	4427	26070	11446 (43.9%)	14343 (55.0%)	281 (1.1%)
UCR	3794	23913	14293 (59.8%)	9421 (39.4%)	199 (0.1%)

Table 3.1: Statistics of topologies generated by different algorithms

peer links in the edge of the Internet, only appear in the BGP paths between their associated ASes, therefore cannot be captured unless we place vantage points in these ASes. In our analysis, we address the incompleteness of topology by adding additional AS links which have been confirmed by other studies. In particular, we choose the data set provided by the latest link discovery study by He *et al.* [47] at UC Riverside, which we call graph *UCR*, and add their newly-found links missing in our topology data.

According to He's study [47], graph UCR is generated based on the data set collected in May 2005. Despite the time difference, we believe most of the links in the old data set still exist today. Table 3.1 presents the basic statistics of graph UCR and 3 other different graphs we generate based on different relationship algorithms, described in Section 3.2.3. Graph CAIDA is directly downloaded from [48] due to the lack of access to the source code of the study [49], Graph SARK and graph Gao are computed based on [45] and [50], respectively, from our collected raw dataset [1]. We discuss these graphs in details in Section 3.2.3. In comparison, graph UCR, slightly smaller than graph SARK and Gao due to its older raw dataset, nevertheless has a higher percentage of peer-peer links most of which were discovered by their proposed techniques. A further comparison of graph UCR with graph Gao shows that 10876 of the 23913 (45.5%) links in the former are missing in the latter. 10847 (99.7%) of these missing links are associated with existing nodes in the latter, indicating that they might be captured if other graph construction techniques (e.g., traceroute in [47]) are used. In Section 3.4, we evaluate how the addition of these missing links affects the overall resilience of the Internet.

[1]Heuristics adopted by the different algorithms do not have definitive relationship inference for certain links in the graph, which results in the little discrepancy between SARK and Gao in Table 3.1.

3.2.3 AS Routing Policy Inference

It is well-known that there are three basic AS relationships [51]: customer-to-provider, peer-to-peer, and sibling relationships. We need to label each link in the topology graph with relationship information required to infer valid, policy-compliant AS paths [52]. Thus, accurate AS relationships are critical to our analysis. Most previous studies on inferring AS relationships [45, 49, 50, 51, 53] are based on heuristics which might not always hold on the real Internet, and therefore, may produce incorrect relationships that directly affect our analysis. For example, a simple test on graphs annotated with AS relationships generated from CAIDA's work [49] reveals the presence of AS routing policy loops.

Although constructing a topology graph matching exactly the current Internet is impossible due to proprietary relationship information, we attempt to create one with maximum accuracy and understand the effect of network topology on routing resilience. A recent study [54] shows that the latest Gao's algorithm [50, 51] and CAIDA algorithm [49] present better accuracy in satisfying "valley-free" [51] policy rule for more AS paths. As such, we first generate a graph using Gao's algorithm with a set of 9 well-known Tier-1 ASes (AS 174, 209, 701, 1239, 2914, 3356, 3549, 3561, 7018) as its initial input. Then we compare the computed graph with graph CAIDA downloaded from [48]. We take the set of AS relationships agreed on by both graphs, which we believe are most likely correct, as the new initial input to re-run Gao's algorithm to produce the graph for our analysis. To ensure valid analysis of the constructed graph, we perform several consistency checks as described below.

- **Connectivity check**: The original topology graph needs to ensure that all AS node pairs have a valid policy path.
- **Tier-1 ISP validity check**: A Tier-1 ISP by definition does not have any providers, nor should their siblings. A Tier-1 ISP's sibling cannot be sibling of another Tier-1 ISP.
- **Path policy consistency check**: There should not be any valid AS path containing policy loops, *e.g.*, a path going from a customer to its provider and eventually returning to the customer serving as the previous hop's *provider*.

Property	Value
# of AS nodes	4427
# of Tier-1 AS nodes	22 (0.5%)
# of Tier-2 AS nodes	2307 (52.1%)
# of Tier-3 AS nodes	1839 (41.5%)
# of Tier-4 AS nodes	254 (5.7%)
# of Tier-5 AS nodes	5 (0.1%)
# of AS links	26070
# of customer-provider links	14343 (55.0%)
# of peer-peer links	11446 (43.9%)
# of sibling links	281 (1.1%)

Table 3.2: Basic statistics of constructed topology

Table 3.2 describes the basic statistics of our constructed topology. We classify the nodes into 5 tiers as follows. We start with the 9 well-known ISPs and classify them and their siblings as Tier-1. Tier-1's immediate customers are then classified as Tier-2. We also ensure all non-Tier-1 providers of these nodes are included in Tier-2. We repeat the same process with the subsequent tiers until all of the nodes are categorized. As we can see, most of the nodes, after the removal of stub AS nodes, are in Tier-2 or Tier-3. Figure 3.1 also illustrates the node degree distribution of the graph. As expected, most networks have only a few providers. About 20% of the networks have at least one peer, which are typically equal-sized networks.

In reality, AS relationships can be much more complicated including per-prefix-based arrangements or combined relationships of transit or provider with customer services [55]. We argue that our simplified approach to constructing the AS-level topology with policy annotations is sufficient for failure analysis, as majority of the prefixes between AS pairs follow one type of policy arrangement. However, we do take care of special exceptions. For example, both Cogent (AS174) and Sprint (AS1239) are well recognized as Tier-1 ISPs, but they do not peer directly as evidenced by lack of AS paths containing links connecting them directly. In reality, Verio (AS2914) provides a transit between their customers. We deal with this case explicitly when computing AS paths.

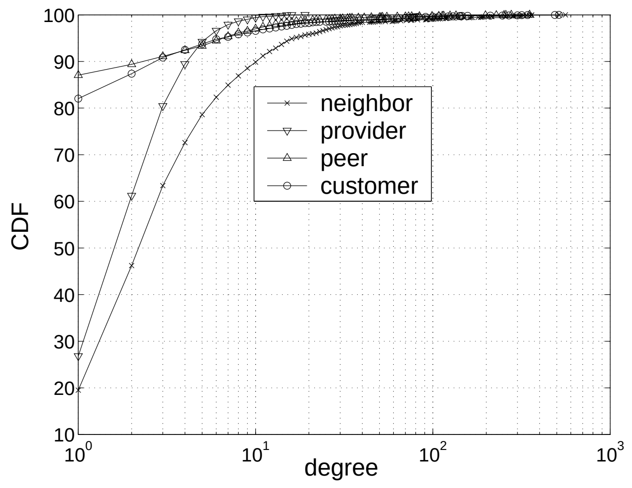

Figure 3.1: CDF of AS node degree based on relationships

Previous link	**Current link**	**Next link**
$\nearrow$	$\nearrow$	$\nearrow$, $\longleftrightarrow$, $\searrow$
$\nearrow$	$\longleftrightarrow$	$\searrow$
$\nearrow$, $\longleftrightarrow$, $\searrow$	$\searrow$	$\searrow$

Table 3.3: Relationship combinations of 3 consecutive links ($\nearrow$: customer-to-provider link, $\longleftrightarrow$: peer-to-peer link, $\searrow$: provider-to-customer link)

3.2.4 AS Relationship Perturbation

As described earlier, no relationship inference algorithm is able to produce a set of AS relationships that exactly matches the actual ones. As a matter of fact, different algorithms could produce vastly different relationship inferences. As shown in Table 3.1, graph SARK has much fewer peer-peer links than graph Gao even though both graphs are computed from the same raw BGP dataset. To justify our evaluation of the Internet resilience, which relies on an accurate AS relationship, we propose a technique to perturb the relationship of certain links to understand the effect of AS relationship distributions on routing resilience.

Each link can be a "peer-peer", "customer-provider" or "provider-customer" link. Here we do not consider perturbation on a sibling link because of its rarity. As such, we have for each link 9 possible combinations of relationship tweaks, based on its relationship before and after the change. First, we discuss how each tweak affects the resilience. Table 3.3 presents all possible combinations of any three consecutive links in a policy-complaint AS path from the perspective

	p-p in SARK	p-c in SARK	c-p in SARK
p-p in Gao	2061	4847	3742
p-c in Gao	1011	9061	359
c-p in Gao	582	296	2723

Table 3.4: Relationship comparison (Gao, SARK)

of the second link (i.e., the link in the middle). Obviously, a peer-peer link is most restricted in finding paths as its previous link has to be a customer-provider link and its next link has to be a provider-customer link. In contrast, a customer-provider or provider-customer link has more options. Changing a peer-peer relationship to a customer-provider or provider-customer relationship thus provides the corresponding link more flexibility in choosing paths, and the overall network resilience is enhanced.

In each of our relationship perturbation, we change the relationship of a number of links. To prevent the tweak of one link from offsetting the tweak of another link, we have to ensure that the tweaks of all of the links are consistent. That is, all of the links involved changing relationships from peer-peer to customer-provider/provider-customer or vice versa. In our current analysis, we only focus on relationship changes between peer-peer and customer-provider/provider-customer. The perturbation between a customer-provider link and a provider-customer link is less realistic and we thus leave it as future work.

Table 3.4 illustrates the comparison between graph Gao and graph SARK. The discrepancies provide candidates for perturbation. Each field indicates the number of links that satisfy the relationship combination. For example, there are 2061 links identified as peer-peer in both graphs and 4847 links identified as peer-peer in graph Gao but as provider-customer in SARK. As shown, there are altogether 8589 peer-peer links in Gao which are customer-provider or provider-customer links in SARK. This set of links is our main focus for the relationship perturbation analysis in Section 3.4. Note that each relationship tweak can only be applied when it does not violate any valley-free rule – the change will not invalidate any AS paths containing the link.

3.2.5 What-if Failure Analysis

Given the inferred AS relationships, we developed an efficient algorithm to construct valid AS-level policy paths between arbitrary AS node pairs. We modify the state-of-the-art algorithm [52] to ensure that the common practice of preference ordering is enforced by preferring customer routes to peer routes and peer routes to provider routes [56]. Figure 3.2 presents the pseudo-code of the algorithm with running time complexity of $O(|V|^3)$. Links in the AS graph are classified as one of the following categories: customer-to-provider link (UP link), provider-to-customer link (DOWN link), and peer link (FLAT link). Accordingly, a path which only follows UP links is called an *uphill* path. Any AS path conforming to BGP policy is of the form of an optional uphill path, followed by zero or one FLAT link, and an optional downhill path. The algorithm starts with the computation of the shortest uphill/downhill paths for all node pairs. Then, it selects from all possible path combinations the shortest path with the preference ordering applied.

Our algorithm is efficient, as we impose an ordering to compute a given AS's provider's routes first both for eliminating unnecessary computation and ensuring consistent routes. Our simulator [57] supports a variety of what-if analyses by deleting links, partitioning an AS node to simulate the various types of failures described in Section 3.3. The simulation tool is designed to be efficient in computing AS paths: all AS-node pairs' policy paths can be computed within 7 minutes with 100 MB memory requirement on a desktop PC with an Intel Pentium 3GHz processor.

3.3 Failure Model

Although the Internet has built-in failure recovery mechanisms through rerouting, there are several real incidents of serious connectivity problems during natural disasters, power outage, misconfigurations, and even intentional attacks [58] against the infrastructure. In Table 3.5, we introduce a failure model capturing the *effect* of network disruption at the global Internet level based on empirical evidence.

As shown in Table 3.5, we categorize the failure scenarios based on the *impact scale*, which we measure by the number of *logical* links affected by the failure. Here, a *logical* link is defined as the peering connection between an AS pair. A logical link might involve several physical links, *e.g.,* two large ISPs peer at multiple geographical locations. We do not explicitly model physical links

```
1. Compute shortest *uphill* paths for all $(src, dst)$ pairs.
   $Dist_{src,dst}$ is the distance of the shortest uphill path
   $Uphill_{src,dst}$ is the shortest uphill path
2. Compute the shortest policy path from $src$ to $dst$
   *function* **shortest_path**(src, dst, $D_{src,dst}$, $P_{src,dst}$)
   # returns $D_{src,dst}$, the length of the shortest path,
   # and $P_{src,dst}$, the shortest path
     **if** $Dist_{dst,src} < \infty$ # choose customer's path
       $D_{src,dst} = Dist_{dst,src}$;
       $P_{src,dst} = Reverse(Uphill_{dst,src})$;
     **else** # choose peer's path
       $D_{src,dst} = \min_p\{Dist_{dst,p} + 1\}$;
       where $p$ is a peer of $src$
       **if** $D_{src,dst} < \infty$
         $P_{src,dst} = (src, p) + Reverse(Uphill_{dst,p})$;
       **else** # choose provider's path
         **foreach** $src$'s provider $m$
           **shortest_path**(m, dst, $D_{m,dst}$, $P_{m,dst}$);
         $D_{src,dst} = \min_m\{D_{m,dst} + 1\}$;
         $P_{src,dst} = (src, m) + P_{m,dst}$;
```

Figure 3.2: Algorithm to compute shortest policy paths for all src-dest pairs

Category:(# of logical links)	Sub-Category	Description
0	Partial peering teardown AS partition	A few but not all of the physical links between two ASes fail Internal failure breaks an AS into a few isolated parts
1	Depeering Teardown of access links	Discontinuation of a peer-to-peer relationship Failure disconnects the customer from its provider
> 1	AS failure Regional failure	An AS disrupts connection with all of its neighboring ASes Failure causes reachability problem for many ASes in a region

Category:(# of logical links)	Sub-Category	Empirical Evidence	Analysis
0	Partial peering teardown AS partition	eBGP session resets Problem in Sprint backbone	 Section 3.4.6
1	Depeering Teardown of access links	Cogent and Level3 depeering NANOG reports	Section 3.4.2 Section 3.4.3
> 1	AS failure Regional failure	UUNet backbone problem Taiwan earthquake, etc	 Section 3.4.5

Table 3.5: Failure model capturing different types of logical link failures.

due to a lack of physical topology information. Based on the number of impacted logical links, we classify failures into three types: no logical link failure, single logical link failure, and multiple logical link failures.

No logical link failure: For reliability and performance reasons, ASes might have more than one single physical link to connect to each other. In particular, if the peering is present at geographically diversified locations, it is be very difficult to completely break the connection between these two ASes. We usually observe the following two types of failures.

- Partial peering teardown: As reported in [59], session reset, due to hardware/software malfunction or maintenance operations, is one of the most frequent routing events in the network. Unless all peering sessions between an AS pair have reset, the two ASes can still maintain their reachability even though traffic performance might be degraded.

- AS partition: Certain physical link failures, occurring inside a single AS, do not cause any damage to its connection to its neighboring ASes. The most severe condition is that the failure breaks the AS into two or more isolated regions, and the networks in different regions can no longer reach each other. We call this type of failure "*AS partition*", as evidenced by a recent event in Sprint backbone [60].

Single logical link failure: A logical link failure indicates the loss of direct connection between the pair of ASes associated with the link. Based on the types of the failed link, we further categorize it into the following two sub-classes.

- Depeering: Depeering occurs when the failure disables the peer-peer link between a pair of ASes. In today's Internet, the largest ISPs (*i.e.*, Tier-1 ASes) establish peer-peer relationships to distribute traffic for their respective customer networks. To gain extra connectivity without increasing financial burden, low-tier ASes also peer with each other. Depeering over a Tier-1 peer-to-peer link can cause significant impact on the Internet as it disrupts the communication between their respective customers and is mostly intentional as evidenced by recent contractual disputes between Cogent and Level3 [61]. In contrast, in the case of lower tier depeering, which is possibly caused by physical damage, misconfiguration, or even intentional link termination, reachability can still be maintained through other provider links with possible performance degradation.

- Teardown of access links: Most networks connect to their providers through the access (*i.e.,*customer-provider) links to reach the rest of the Internet. A failure on such access links can severely disrupt the customer's reachability. This type of failure might be one of the most common link failures, as evidenced by the frequent reports in NANOG [62].

Multiple logical link failures: This type breaks multiple logical links, thus causing much more severe impact.

- AS failures: one particular scenario, we denote as "AS failure", occurs when all the logical links between an AS and its neighbors fail, indicating that the corresponding AS is unable to originate or forward any traffic. This can be caused by hardware malfunction or misconfiguration inside the failed AS. For instance, UUNet backbone problems [63], despite its undisclosed causes, resulted in significant network outages.

- Regional failures: are often caused by natural disaster and intentional attacks, resulting in multiple logical link or AS failures in the affected region. In addition to local networks in the region, other parts of the Internet whose traffic traverses the region are also impacted. Well-known examples include 911 attack [19], Hurricane Katrina [64], as well as the recent Taiwan earthquake [21].

As evidenced by various real events, the Internet is susceptible to certain types of failures, especially when critical nodes (UUNet problem) or links (Cogent and Level3 depeering) are involved. In Section 3.4, we use our simulation tool to conduct a more systematic evaluation of the impact of different types of failure on the Internet.

3.3.1 Case Study: Taiwan Earthquake

Given the known disruption to the Internet due to the recent Taiwan earthquake [21], we perform a more detailed study of its impacts in the region on the third day after the earthquake happened. The earthquake occurred in December 2006 near Taiwan, damaging several undersea cable systems in Asia. Many networks in Asia were affected, causing degraded performance, and network connectivity problems in Asia were globally felt for weeks.

We first collected BGP data for that period of time from RouteViews and RIPE which captures the earthquake effects based on the number of ASes or prefixes that experience path changes (or even complete withdrawals). In addition, given that the effect of the earthquake was relatively long-lasting due to the long repair time, we augment our analysis with traceroute probes. In particular,

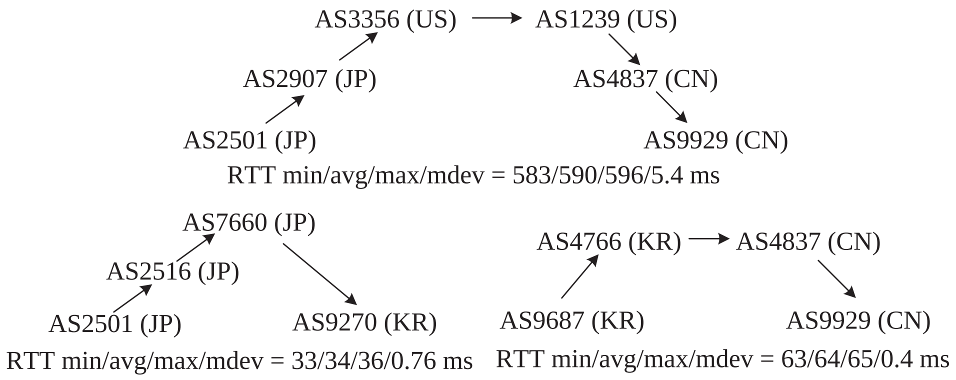

Figure 3.3: Top route is inefficient but can be improved by composing two bottom routes.

we probe from PlanetLab hosts [65] located in several Asian countries and other areas of interest: China, Singapore, Taiwan, Japan, South Korea, US, and Australia. The goal is to understand possibly abnormal paths with long delays and to locate the bottleneck causing the slowdown.

We summarize our findings. Most affected prefixes belong to networks in Asian countries around the earthquake region. For example, 78-83% of the 232 prefixes announced from a large China backbone network were affected across 35 vantage points. Most of the withdrawn prefixes were re-announced about 2 to 3 hours later. We found that many affected networks announced their prefixes through their backup providers. For example, before the event all the vantage points went through AS1239 to reach China 169 backbone (AS4837). After the earthquake, backup paths through networks such as AS3320, AS7018, and AS1239 are used. We identified several AS-level links experiencing problems. For example, before the event, all the vantage points traversed AS1239 to reach a Singapore network, AS4657. After the earthquake, they instead choose other ASes such as AS209 and AS2914.

By actively performing traceroute probing from 8 PlanetLab nodes in 8 distinct Asian countries, we found that interestingly, traffic between some network prefixes in Asia are routed via other remote continents during the period after the earthquake. For example, The Taiwan Academic Network to China Netcom were routed from Taiwan to NYC before reaching China Netcom. The roundtrip delays can exceed 550ms due to the long distance and congestion. During normal period though the AS level path is the same, packets are routed within the east pacific area. As shown in Figure 3.3, we found that from the PlanetLab node in Japan to a China commercial network, the

	AU2	CN2	HK2	JP2	KR2	SG2	TW2	US2
AU	11	657	433	271	335	392	304	229
CN	570	150	41	446	318	83	286	475
HK	288	219	2	127	137	40	446	337
JP	152	450	117	21	44	94	137	169
KR	287	203	655	40	5	468	378	172
SG	391	412	37	208	355	90	360	267
TW	270	559	456	32	280	471	1	182
US	242	205	251	190	194	296	188	8

Table 3.6: Latency matrix among Asian countries in msec (from educational to commercial networks)

path goes through the US, taking a long time to travel over excessive physical distances. However, two networks in South Korea have direct connections to both Japan and China networks. Hence, if the networks in Korea can provide temporary transit services for both China and Japan, we obtain an overlay path through Korea with a much shorter physical distance.

To generalize our analysis, we obtained a latency matrix among Asian countries and the US from educational to commercial networks shown in Tables 3.6. Based on this, we identify that at least 40% of paths with long delays can be significantly improved by traversing a third network. The best improvement reduces latencies from 655ms to only around 157ms (from KR to HK2 when asking JP to provide the transit service). More details of the study are presented in [57].

3.4 Impact Analysis of Failures

We now analyze each failure type to understand the impact at the Internet scale. Note that we focus on *logical* link failures only, which corresponds to failures of one or more physical links. Such failures are not unlikely as evidenced in the past. In what follows, unless otherwise specified, a link implicitly means a logical link, and a node refers to an AS.

3.4.1 Evaluation Metrics

A failure disrupts the traffic that traverses the failed network component and the traffic has to be rerouted via a different path to reach its destination. To quantify failure impact, we define the following two metrics:

- *Reachability impact*: In the worst-case failure scenario, no alternative path can be located between the source and the destination. We define two types of reachability impact: the *absolute* reachability impact R^{abs} and the *relative* reachability impact R^{rlt}. R^{abs} is the number of AS pairs that lose reachability to each other during the failure. In addition, We define R^{rlt} as the percentage of disconnected AS pairs over the maximum number of AS pairs that could possibly lose reachability.

- *Traffic impact*: After the failure, the traffic that used to traverse the old failed link is shifted onto the new paths. The shifted traffic could lead to serious network congestion. Due to the lack of accurate information on actual traffic distribution among ASes, we instead estimate the amount of traffic over a certain link as the number of the shortest policy-compliant paths that traverse the link, denoted as *link degree* D. We compute the link degree D of all links before and after the failure, and estimate the effects of traffic shift by calculating these 3 metrics: (1) the *maximum* increase of D among all links T^{abs}, (2) the *relative* increase of D of this link T^{rlt}, and (3) the maximum relative increase in D of the failed link T^{pct}. Suppose the link A is failed, and most of its traffic is shifted to link B. The three metrics are computed as follows.

$$T^{abs} = D_B^{new} - D_B^{old}, T^{rlt} = \frac{T^{abs}}{D_B^{old}}, T^{pct} = \frac{T^{abs}}{D_A^{old}} \tag{3.1}$$

  The first two quantify the impact of traffic shift on individual links while T^{pct} captures the evenness of re-distributed traffic for the failed link. Although the link degree cannot exactly quantify the traffic impact in each failure because of the uneven traffic distribution in the Internet, it, which computes the increased number of AS paths that traverse each link, provides a good estimate on the amount of shifted traffic.

3.4.2 Depeering

Today's Internet core consists of a group of large ISPs known as Tier-1 ASes which are the top service providers. Their customers can reach each other via the peer-peer links among the Tier-1 ASes, so these peering links are critical to maintaining the Internet connectivity. In this section, we

Tier-1 AS	174	209	701	1239	2914	3356	3549	3561	7018
# of single-homed customers without stubs	16	13	9	13	11	30	15	10	9
# of single-homed customers with stubs	193	229	45	47	43	162	53	55	49

Table 3.7: Number of single-homed customers for Tier-1 ASes

AS	174	209	701	1239	2914	3356	3549	3561
174	/	/	/	/	/	/	/	/
209	100	/	/	/	/	/	/	/
701	87	91	/	/	/	/	/	/
1239	79	91	85	/	/	/	/	/
2914	100	93	100	85	/	/	/	/
3356	100	95	100	85	100	/	/	/
3549	82	99	82	85	100	87	/	/
3561	87	92	100	89	100	100	100	/
7018	92	100	92	100	92	92	92	100

Table 3.8: R^{rlt} (%) for each Tier-1 depeering

analyze the effects of peering (particularly the Tier-1 peering) link failures on network reachability and traffic shift.

Table 3.7 presents the number of single-homed customers with and without the stub ASes for each Tier-1 AS, where *single-homed* refers to customers that can only reach only one Tier-1 AS through uphill paths. If all the physical peering links between two Tier-1 ASes stop working, *i.e.*, a logical link failure, their respective single-homed customers can only reach each other using the lower-tier peering links.

We first analyze how each Tier-1 depeering affects loss of network reachability due to unreachable AS pairs of single-homed ASes of the Tier-1 ASes involved. Because of the rich connectivitity in the Internet, some pairs of the single-homed ASes of the depeered Tier-1 can still reach each other via low-tier peering links. We use the relative reachability impact $R^{rlt}_{i,j}$ to quantify the impact,

$$R^{rlt}_{i,j} = \frac{\#\ of\ disconnected\ pairs}{1/2 \times S_i \times S_j}, \tag{3.2}$$

where S_i and S_j indicate the number of single-homed ASes for the two depeered Tier-1 ASes i and j. Table 3.8 presents the results for our graph without stub ASes. Tier-1 depeering disrupts connections among most single-homed customers. Overall, 89.2% of pairs of Tier-1 ISP's single-homed customers suffer from reachability loss, while the remaining pairs manage to detour using lower-tier peers or siblings. If we consider the stub ASes, 298493 (93.7%) out of 318562 single-homed AS pairs lose reachability.

We examine pairs of single-homed customers that remain connected after depeering. Among all 744 connected pairs, 86% of them traverse peer-peer links, and the remaining 14% have common low-tier providers.

Second, we investigate the effects of Tier-1 depeering on traffic shift. We observed, on average, the maximum traffic increase of a link, *i.e.,* T^{abs} is 3040 (with maximum of 11454), which corresponds to 22% (with maximum of 62%) of the traffic of the depeered link (*i.e.,* T^{pct}) being shifted. Our results also show the relative traffic increase T^{rlt} could reach up to 237% with an average increase of 61%, indicating that the traffic shift might impose a serious burden on certain links.

We also analyze depeering of lower-tier peering links. Even though they do not impact network reachability due to the ability to use Tier-1s to reach each other, we examine the traffic impact. We pick 20 most utilized non-Tier-1 peer-to-peer links, and simulate the path changes after the failure of each link. Our results show that the average maximum traffic increase T^{abs} is 14810, and the corresponding T^{pct} and T^{rlt} are 35% and 379%, respectively, indicating that lower-tier peering links can also introduce significant traffic disruption.

A. Effects of Missing Links

As we discussed in Section 3.2.2, our topology graph, constructed solely from BGP measurement data, cannot capture all the links in the Internet. We add the newly-discovered links in graph UCR to examine how it affects the simulation results.

A total of 10847 links are added, containing 8059 (74.3%) peer-peer links, 2753 (25.4%) customer-provider links, and 35 (0.3%) sibling links. For comparison purposes, we use the same set of single-homed ASes in our analysis. 5892 (85.5%) pairs of ASes experiencing loss of reachability in the new graph, compared to 6143 (89.2%) pairs of ASes in the old graph. As expected, adding new links slightly improves the resilience under Tier-1 depeering as the new links can be used to locate alternative paths.

# of perturbed links	0	2k	4k	6k	8k
% of disconnected ASes	89.2	88.6	87.9	87.2	86.3

Table 3.9: Effects of perturbing relationship.

B. Effects of Relationship Perturbation

Next, we evaluate how perturbing the relationship described in Section 3.2.4 affects the analysis results. We have a candidate set of 8589 peer-peer links which can be changed to customer-provider links. In our evaluation, we test 4 different scenarios in which 2000, 4000, 6000, and 8000 peer-peer links in the candidate set are randomly selected and changed to customer-provider or provider-customer links. For each test scenario, we randomly generate 5 different graphs.

For comparison purposes, we consider the same set of single-homed ASes and evaluate how the perturbation affects the connectivity between any pair of these ASes. Table 3.9 presents the percentage of single-homed AS pairs that lose reachability under different scenarios. As shown in the table, perturbing the relationship slightly improves the resilience of the network as the perturbed provider-customer links either make single-homed ASes become multi-homed or provide better lower-tier connectivity. *The quite limited improvement also indicate that these single-homed customers have very limited access links to reach Tier-1 ASes and uninformed, random relationship perturbation does not improve their routing resilience much.*

To summarize, Tier-1 depeering disrupts the reachability of only a small number of ASes that are single-homed to the affected Tier-1 ASes, nevertheless, these affected ASes experience severe damage as they can no longer reach 89% of the rest of the ASes.

3.4.3 Teardown of Access Links

After the analysis of failures of peer-peer links, we now study how the failure of customer-provider links (also known as *access link*s), which counts for 77% of all AS links in the Internet, affects the network reachability. The robustness of connectivity of an AS can be captured by the similarity of its paths reaching the *Tier-1* ASes, given that Tier-1 ISPs are so richly connected; thus, reaching them is very important. For example, in the Tier-1 depeering analysis, ASes with uphill paths to multiple Tier-1 ASes can survive the depeering disruption without losing reachability to other ASes.

Path similarity can be defined as the number of commonly-shared links among all the paths under consideration. In particular, nonzero path similarity means that failing *a single link* can disrupt reachability. For instance, similarity of 2 implies that there exists two commonly shared links among all possible paths; therefore breaking any of the two links will create disruption.

We now describe how to calculate the path similarity of each AS to the set of all Tier-1 ASes to evaluate the robustness of the connectivity and to identify critical links. We first transform this problem into a max-flow-min-cut problem [66]. We solve the minimum-cut problem by using an approach based on the “push-relabel” method [66] and then present our analysis for scenarios with the BGP policy imposed and also those without policy restrictions. Moreover, we study the impact of failures of commonly-shared links which tend to be critical for the network.

Since our focus is on finding cases of nonzero path similarity, we transform the problem into a max-flow-min-cut problem by assigning a capacity of 1 for every link in the graph. The solution identifies the maximum flow that can be transferred between a source s and a sink t. Because each link has a capacity of 1, once we have a solution with a maximum flow value of 1, there has to be at least one link shared by all paths between s and t.

In our analysis, we have one source and multiple sinks. The source can be any non-Tier-1 AS while the multiple sinks are the Tier-1 ASes. We create a supersink t and add a directed link from each Tier-1 AS to t with a capacity value of ∞. We perform the analysis for both conditions of BGP policy constrained path selection and no policy restrictions. For the latter, we transform our topology into an undirected graph. For the former, since we consider the uphill paths of each non-Tier-1 AS to Tier-1 ASes, which do not contain any peer-peer links, we remove all peer-to-peer links from the topology, while keeping each customer-to-provider link as a directed link pointing from the customer to the provider, and making each sibling link undirected. All links in the converted graph have capacity value of 1 except for the links to the supersink.

Under no policy restrictions, 703 (15.9%) out of 4418 non-Tier-1 ASes have a min-cut value of one and can thus be disconnected from the network by removing only one of the commonly-shared links. This implies that *despite apparent physical redundancy, a fairly large number of networks on the Internet are vulnerable to significant reachability disruption caused by a single access link failure even without policy restrictions.*

```
function find_path($src$, $dst$, $last$, $link\_set$)
# if returns TRUE, paths exist between $src$ and $dst$;
# $link\_set$ is the set of links shared by these paths
  if ($src$ = $dst$)
    ret = TRUE; $link\_set$ = $\{(last, dst)\}$
  else
    $S$ = $\{$all links$\}$; ret = FALSE; # initialize S and ret
    foreach $x \in \{src$'s providers or siblings$\}$
      if (find_path($x$, $dst$, $src$, $S_x$) = TRUE)
        $S = S \cap S_x$; ret = TRUE;
    $link\_set = S \cup \{(last, src)\}$;
  return ret;
```

Figure 3.4: Algorithm to locate shared links among all paths from src to dst.

# of shared links	0	1	2	3	4
percentage	78.3	18.3	3.1	0.3	0.02

Table 3.10: Number of commonly-shared links.

# of nodes	1	2	3	4	5	> 5
percentage	92.7	4.5	1.6	0.1	0.3	0.7

Table 3.11: Number of ASes sharing the same critical link.

Under BGP policy restrictions, 958 (21.7%) of 4418 ASes have a min-cut value of 1, and about 255 (6%) of the ASes are susceptible to single link failures even though they have physical connectivity. This indicates *BGP policies severely limit network reachability under failures, and relaxing policies can help alleviate the failure impact.*

Recall that stub ASes excluded from our topology graph tend to have even more limited connectivity due to being single-homed. In our graph, we exclude 21226 stub ASes, 7363 (34.7%) of which have only one provider and are thus subject to a single access link failure. Considering the stub ASes, at least 8321 (32.4%) of the ASes are vulnerable to single access link failure.

The default s-t max-flow-min-cut solution only generates one possible cut. We develop a recursive algorithm for finding the set of all commonly-shared links among all possible paths between a given non-Tier-1 AS and the set of Tier-1 ASes, shown in Figure 3.4. By remembering partial results, the running time complexity of this algorithm is $O(|V| + |E|)$.

Table 3.10 shows the percentage of the number of shared links from any non-Tier-1 AS to all Tier-1 ASes. Most of the ASes that share link(s) have only 1 common link while few nodes share as many as 4 links to reach Tier-1 ASes. This implies that *the attack of a randomly selected link is unlikely to significantly disable the targeted AS's connectivity from other networks.* We also collect the statistics on the links that are commonly shared by any of these ASes.

Table 3.11 presents statistics on the number of AS nodes that share the same critical link. Removing each of these links disrupts the connectivity of all of the ASes that share the link. More than 90% of the links are shared by only one AS while few links are shared by more than 10 ASes to reach the set of Tier-1 ASes. This indicates *a single logical failure has a limited scale of impact as ASes rarely share a common critical access link.*

To capture the impact of removing shared links, we study failure scenarios in which any of the 20 most shared links is disabled. We estimate the impact by using previously defined metrics. Upon failure, the affected AS(es) can no longer reach the Tier-1 ASes and their reachability to other networks solely relies on their alternate lower-tier connectivities. We use the relative reachability impact R_l^{rlt} for failed link l as our metric,

$$R_l^{rlt} = \frac{\#\, of\ disconnected\ pairs}{1/2 \times S_l \times (S - S_l)}, \tag{3.3}$$

in which S_l and S indicate the number of ASes that share the failed link l and the total number of ASes in the graph, respectively. For the 20 scenarios analyzed, our results show that the average value of R^{rlt} is 73.0% with standard deviation of 17.1%. *Failures of shared access links disrupt most of the reachability for ASes that share the removed links.* In the few cases when reachability is not impacted, the corresponding pairs of ASes use low tier links similar to depeering to route around the failed link.

For the traffic impact, the maximum increase T^{abs} among the 20 failures is 53179, accounting for 50.3% of the total traffic shift, *i.e.*, T^{pct}.

A. Effects of Missing Links

Similar to the depeering analysis, we evaluate how the addition of new links learned from graph UCR affects our conclusions. With added links, our results show that 678 (15.3%) of the

# of perturbed links	0	2k	4k	6k	8k
# of ASes with min-cut 1	958	928.6	901.3	873.5	848.9

Table 3.12: Perturbing relationships: improved resilience.

ASes have min-cut value of 1 under no policy restriction showing an increase of 25 (0.6%) ASes no longer sharing common links. Under policy restrictions, however, 956 (21.6%) of the ASes have min-cut value of 1, *i.e.*,, only 2 (0.05%) additional ASes becomes insusceptible to single link failures with additional links. For the failures of the same 20 shared links, the average of R^{rlt} is 68.7% with standard deviation of 14.3%.

We can conclude that *although the addition of new links increases the physical connectivity of networks, it only slightly improves the resilience for access link failures as the added links, most of which are peer-peer links, have a limited access to reach the affected ASes.*

B. Effects of Relationship Perturbation

We next discuss how relationship perturbation affects the min-cut analysis results. Similarly, we simulate failures on 4 different graphs in which we change 2000, 4000, 6000, and 8000 peer-peer links in the candidate set to customer-provider/provider-customer links. We randomly generate 5 tests for each scenario. We focus on min-cut analysis under BGP policy restrictions.

Table 3.12 presents the min-cut analysis results for 4 relationship perturbation scenarios. *Changing peer-peer links to customer-provider/provider-customer links improves the overall network resilience as the perturbed links provides ASes extra flexibility in choosing paths to other networks.*

To summarize, despite the apparent physical redundancy, a surprisingly large number of ASes are vulnerable to a single access link failure, which we believe is the most common failure in today's Internet. Even worse, BGP policies severely further limit the network resilience under failure: about 35% of the ASes can be disconnected from most of the rest of the network by a single link failure.

3.4.4 Failure of Heavily-used Links

Shared links to reach Tier-1 ASes can be considered as one type of *critical* links. We also analyze the impact of failures of another type – links used by many networks or heavily-utilized

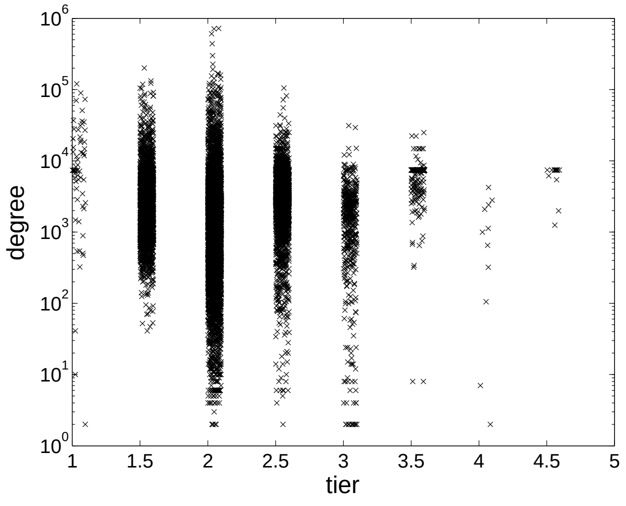

Figure 3.5: Link degree vs. link tier.

links based on their topological location.

Figure 3.5 is a scatter plot of the link degree vs. link tier. *Link tier* is calculated as the average of tier values of the two ASes of the link. For example, if the link is between a Tier-1 AS and a Tier-2 AS, the link tier is 1.5. *Link degree* D, as defined in Section 3.4.1, is the number of AS pairs traversing the link. As shown in the figure, the most heavily-used links are within Tier-2. This is expected as core links carrying significant amount of Internet traffic have high link degrees.

In our simulation, we select 20 most heavily utilized links as failure targets, excluding Tier-1 peer-to-peer links which have been studied in Section 3.4.2. These 20 links either reside in Tier 2 or connect between Tier-1 and Tier-2 ASes and are traversed by 0.9% up to 5.2% of paths between all AS pairs. In each simulation run, we remove one of these 20 links and estimate the failure impact. In particular, we examine how those AS pairs that used to traverse the broken link fail over to new paths. Our analysis shows that 18 out of 20 failures do not disrupt reachability between any AS pairs. In fact, the two cases that impact reachability involve two shared links as evaluated in Section 3.4.3.

For the 20 failures studied, the maximum T^{abs} is 113,277 with an the average of T^{abs} 64,234 while the maximum T^{pct} is 77.3% with the average of T^{pct} 38.0%. These values indicate significant, uneven traffic re-distribution that may require traffic engineering to reduce potential congestion.

3.4.5 Regional Failures

We now present simulation-based analysis of a particular regional failure scenario. We first describe the method to determine the set of affected ASes and links before presenting the analysis on the failure impact.

Motivated by several real incidents such as the 9/11 attack and the 2003 Northeast blackout, our regional failure simulates the scenario when all ASes and links traversing New York City (NYC) are broken. Unlike the previous scenarios that focus on single link failures, regional failures usually affect multiple links and tend to have larger impact.

We first use NetGeo [67] to approximately identify the set of ASes and links that can be affected by events in NYC. NetGeo provides a set of geographic locations for each AS. Because our analysis is based on the AS-level granularity, we select ASes located in NYC only and thus ignore partial AS failure for simplicity. To identify relevant links, we first choose links whose both end points share a single common location in NYC. In addition, NYC might also be critical to links with a single end point in NYC. For example, we observe that South African ISPs connect to New York as their main exchange point to the rest of the Internet even though NetGeo indicates they only reside in South Africa.

To capture such long-haul links connecting NYC to a remote region, we perform traceroute from PlanetLab hosts located different foreign countries to 35 PlanetLab ASes located near NYC. If traceroute results exhibit any stops in NYC, we include the corresponding AS links. Due to limited probing, our analysis may miss some links impacted by the failure. A total of 268 ASes and 106 links (56 of them are customer-to-provider links; the remaining are peer-to-peer links) are selected to fail concurrently in our simulation.

Our simulation shows that this example regional failure disrupts the reachability between 38,103 AS pairs, which mainly involve only 12 ASes, which we separate into 2 sets according to their failure patterns.

Case 1: One AS (located in South Africa) used to have 2 providers and 2 peers. The failure disabled its links to both of its providers, leaving it with only 2 peers to connect to the rest of the Internet.

Case 2: This set includes 11 ASes located in one of the European countries. Similar to the previous case, the failure caused breakage of their provider link(s). However, these ASes do not have peers,

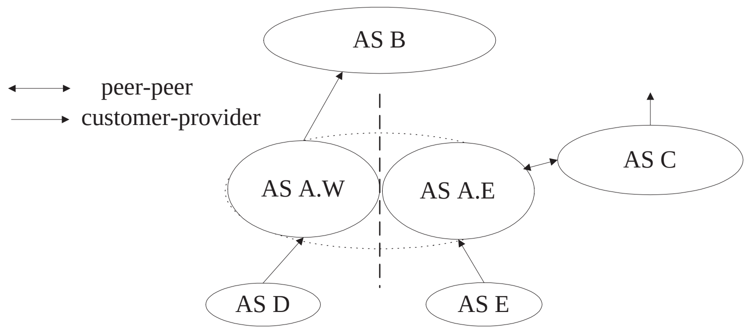

Figure 3.6: An example of AS partition

leaving them isolated from the rest of the Internet due to the failure.

In both cases, the affected ASes experience the failure of its shared access link(s) as discussed in Section 3.4.3 as their paths to Tier-1 ASes are disrupted. Regional failures cannot cause Tier-1 depeering due to their rich geographic diverse peering. *Most damage caused by the regional failures is due to the failure of critical access links.*

We also evaluate the potential impact of the failure on traffic caused by traffic shift from paths that used to traverse the affected region. This imposes extra traffic load on links in other regions. we found T^{abs} to be as high as 31,781.

3.4.6 AS Partitions

In this section, we examine scenarios when failures break an AS into two or more isolated parts and disrupt connectivity among these AS partitions. We first describe our analysis method before presenting the results.

First, we use an example in Figure 3.6 to illustrate how an AS partition disrupts reachability. AS A is partitioned into two parts, $A.E$ and $A.W$. A direct effect is that the communication between its separate parts is disrupted as $A.E$ and $A.W$ cannot reach each other unless their neighbors can provide extra connectivity to bypass the failure. (Special configuration, *e.g.,* tunneling, needs to be set as the neighbors cannot use the AS number to distinguish the partitions.) As described previously, the reachability resilience of an AS is indicated by the diversity of its uphill paths to the Tier-1 ASes. No reachability will be disrupted unless one of its partitions, AS $A.E$ as well as its single-homed customer E, loses connection to its only provider AS B. As such, *the AS* *partition becomes equivalent to the failure of an access link as discussed in Section 3.4.3.* Note that even though AS C in the example can no longer reach $A.W$, it can still reach $A.W$ through its provider(s).

In our analysis, we simulate a special case of AS partition in which a Tier-1 AS is separated into two parts. Due to the lack of detailed AS specific geographical information such as peering location, it is very challenging to model a network partition accurately. Since a Tier-1 AS spans over most of the country, we simulate the partition by breaking the AS into 2 parts: east region and west region. Based on its geographical presence from NetGeo data, we classify each neighboring AS of the target Tier-1 AS into 3 types: “east neighbor”, ”west neighbor” and “other neighbor” which resides in both regions. The failure only affects east or west neighbors. The Tier-1 AS in our simulation contains 617 AS neighbors, 62 of which in the east and 234 in the west.

In the simulation, we transform the old Tier-1 AS into two pseudo ASes. The east/west neighbors connects to only one of these new ASes while the rest of the neighbors have links to both ASes. Because Tier-1 ASes peer at many locations, the partition does not break any of the peering links. Failure only affects the communication between the single-homed ASes in the east and those in the west. To estimate the reachability impact, we choose R^{rlt} as the metric and S_i and S_j are the number of single-homed customers in east and west, respectively. Our results show that the partition disrupts 118 pairs of ASes with R^{rlt} 87.4%.

3.5 Related Work

Several previous work [68, 69] on understanding the resilience of the Internet to faults are based on a simplified topology graph without policy restrictions and thus may draw incomplete conclusions. They also do not provide suggestions on improving failure resilience. We build on previous work [70] on analyzing how location of link failures affect the Internet and extend it to realistic topologies with routing policies as well as more general failure models. Our work also makes contribution in developing more accurate Internet routing models by focusing on the structure of the network. We take a different approach from recent work [71] by modeling routing decisions based on policies while accommodating multiple paths chosen by a single AS. Unlike previous studies focusing on obtaining complete AS topologies [47, 46], our focus is understanding how the topological structural properties affect routing resilience to failures.

In the area of understanding network resilience, a common method for analyzing network resilience is to compute the number of node or link disjoint paths between any pair of ASes, *i.e.*, path diversity of the Internet. Teixeira *et al.* [72] studied the path diversity problem both inside an AS (Sprint network) and across multiple ASes based on the CAIDA topology. In comparison, we present a more systematic evaluation of the resilience problem based on more complete and accurate topology data. Previous study by Erlebach *et al.* [73] also proposed using the min-cut analysis to compute the maximum disjoint paths between a pair of ASes, which is shown to be NP-hard. Instead of developing approximation algorithm, our analysis simplifies the path diversity problem by precisely locating critical links between an AS and the set of Tier-1 ASes. Our technique is shown to be efficient and capable of identifying weakness in the Internet.

3.6 Concluding Remarks

In this chapter, we have presented a comprehensive framework to analyze the resilience of Internet routing to common types of failures captured by our failure model which is developed based on empirical analysis. Our efficient simulation tool enables us to study how network topologies and routing policies influence network failure resilience measured using basic metrics of network reachability and traffic impact.

We summarize our main results of analyzing routing resilience to failures. (i) Tier-1 depeering, despite its infrequent occurrence, disrupts most of the reachability, *i.e.*, 94%, between the single-homed customer ASes of the affected Tier-1 ASes. (ii) Most of the reachability damage in today's Internet is caused by failures of the *critical access links*, which are traversed by all possible paths from the affected AS(es) to the rest of the Internet. We found out that 32% of the ASes are vulnerable to this type of the failure, most of which we believe is due to the nature of single-homing. Today's Internet might not be as resilient as we thought. (iii) BGP policy limits the ASes' option in selecting paths to reach other ASes, an additional 255 (6%) non-stub ASes can be disrupted by a single link failure even though the physical connectivity might be available to bypass the failure. (iv) Traffic is not evenly re-distributed during the failure and results indicate that more than 80% of the traffic over the failed link can be shifted to another link. (v) Adding extra links into the graph and perturbing relationship on certain links slightly improves the resilience of the network. The fundamental conclusion drawn above, nevertheless, stays the same.

Given our simulation-based failure analysis, we make the following observations to help enhance routing resilience: (i) We need extra resources (e.g., multi-homing) to be deployed around the weak points of the network. Approaches like sharing resources among neighboring ASes [74] can also be used. (ii) Based on the observation that policy further restricts path selection, other techniques to better utilize physical resources can also improve the resilience during failures, *e.g.*, selectively relaxing BGP policy restrictions. (iii) From our earthquake study, we learn that for some cases, even though reachability might not be affected, the performance will be severely degraded. (iv) Regional failures such as 911 has more global impact due to long-haul links connecting to remote regions.

To our best knowledge, this is the first detailed study of the impact of significant but realistic failures on the Internet, using both reachability and increase in traffic paths along links which reflect the impact on application performance. Our study reveals the vulnerability of the Internet routing through detailed data analysis of existing well-known failure events to provide insights into the derivation of solutions. The critical links identified by our simulation analysis tool can benefit the design of both short-term mitigation responses as well as other long-term improvements.

CHAPTER 4

Improving Internet Routing Resilience Using Dynamic Negotiation

In the previous chapter, we propose a framework that systematically analyzes how the current Internet routing system reacts to various types of large-scale failures. We demonstrate how restrictions imposed by routing policies can prevent network reachability under various failures, thus disallowing routing to fully take advantage of the underlying network physical redundancy. In this chapter, we improve the robustness of the Internet interdomain routing by allowing ASes to relax the policy restrictions when needed so that their surrounding physical redundancy can be exploited.

4.1 Introduction

On average, routing on today's Internet works reasonably well, maintaining reachability for most networks and achieving good performance across most network paths. However, from our study on the Internet's resilience to failures in Chapter 3, certain network components are vulnerable to two types of failures which can be caused by realistic events, such as the 911 terrorists attack [19], the Northeast blackout [20], the recent Taiwan earthquake [21], and the Middle East undersea cable cut [75]. The core of today's Internet consists of a group of large ISPs known as *Tier-1* ASes. *Tier-1 depeering*, in which the mutual transit service between a pair of Tier-1 ASes is terminated, handicaps the Internet [61] as the large number of single-homed customer ASes of the affected Tier-1s can no longer communicate with each other. We also found that certain ASes have to traverse a common set of links to reach the rest of the Internet. The failure of these *critical* links disconnects the corresponding AS(es) completely from the Internet. Surprisingly, however, we found sufficient redundancy of physical connectivity in the proximity of the failed components, which can be utilized for service restoration if the interdomain policy is not enforced.

Recently, various techniques have been proposed to improve the Internet's resilience to failures by either enhancing the availability of routing information or expediting routing convergence. Route deflection [76] and BGP splicing [77] allow packets to be forwarded over other than just the shortest paths. MIRO [78] proposes ASes to negotiate on the set of routes to be exchanged so that each AS may acquire information on extra routes to circumvent failed ASes. In R-BGP [79], each AS precomputes a backup path for each destination and the routing convergence after a failure will not affect the delivery of packets. Unfortunately, most of these techniques assume whatever path taken satisfies the current interdomain policy, making them unable to handle the above-described situation in which no policy-compliant path can be found. In this chapter, we propose a new mechanism called *Dynamic Routing Negotiation* (DRN) that can overcome currently prevalent policy restrictions to significantly enhance the Internet routing resilience and better utilize the redundant connectivity in network topology. In DRN, when an AS can no longer reach certain destinations after a failure, it negotiates with its neighbor ASes to temporarily relax the normal policy restrictions so that more paths may be identified and utilized to circumvent the failure. Our in-depth simulation on realistic Internet topologies has shown that, by relaxing interdomain polices between peers alone, DRN recovers 100% of service disruptions caused by Tier-1 depeering and 63% of those caused by critical link failures.

The chapter is organized as follows. Section 4.2 provides a brief background of BGP and a summary of our early related study and presents an empirical study that motivates this work. Section 4.3 details the DRN mechanism, and discusses some design parameters that are important to the DRN's performance. Section 4.4 evaluates the performance of DRN via extensive simulation on realistic Internet topologies. Section 4.5 discusses the related work, and the chapter concludes with Section 4.6.

4.2 Background and Motivation

This section first provides a brief background of interdomain routing and a summary of previous work on the structural weaknesses of the Internet in the face of failures of network routers and links. Next, it presents an empirical study of the Taiwan earthquake and the motivation for our proposed DRN. To put DRN in perspective, we provide a taxonomy of the recently-proposed techniques for improving the Internet's resilience to failures.

4.2.1 BGP and Achilles Heel of the Internet

The Internet consists of thousands of ASes operated by many different administrative entities, such as Internet Service Providers (ISPs), companies and universities. Interdomain routing in the Internet is coordinated by the Border Gateway Protocol (BGP) [80]. ASes use BGP to exchange reachability information—*i.e.*, the list of ASes along the path to the destination—with each other to provide global connectivity. One distinct feature of the interdomain routing protocol is that, when multiple paths to a destination are available, ASes use a combination of local policy, AS-path length, as well as other local constraints to select the best path. The commercial contractual relationship with an adjacent AS is one of the most important factors for a local policy. Typical inter-AS relationships include *customer-provider*, *peer-peer*, and *sibling*. In the first type of relationship, a customer pays its provider for connectivity to and from the rest of the Internet. In the peer-peer relationship, ASes benefit from direct access to each of their respective customers free of charge, while sibling allows friendly or related ASes to provide connectivity to the rest of the Internet for each other.

Under the interdomain routing policy, physical connectivity does not imply reachability because the policy imposes restrictions on the selection of routing paths. For example, the customer does not transit traffic between two of its providers. Peers transit traffic only for their respective customers, but not their providers or other peers. In summary, an AS *selectively* provides transit service for its neighboring ASes and AS paths in the Internet often exhibit the "*valley-free*" property [51].

In Chapter 3, we developed a methodology to identify the interdomain structural weaknesses of the Internet. Our major findings are summarized as follows.

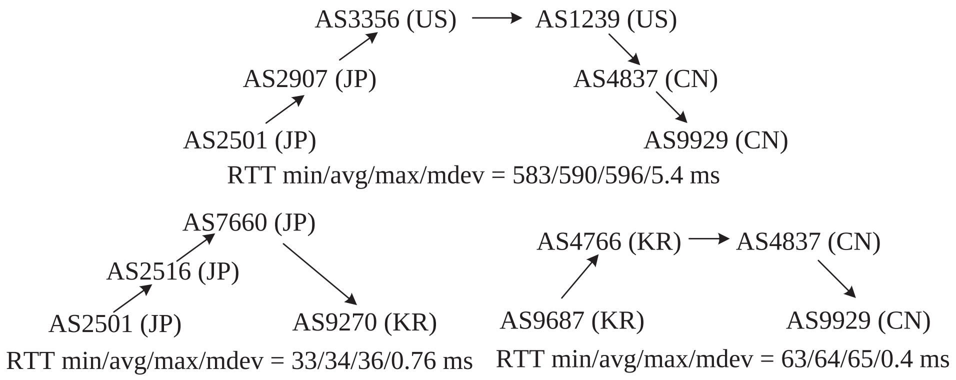

Figure 4.1: Top route is inefficient but can be improved by composing two bottom routes.

- Tier-1 depeering, often due to administrative reasons, disrupts 94% of connections between single-homed customer ASes of the affected Tier-1 ASes.

- Certain customer-provider links are *critical* to ASes that must traverse to reach the rest of the Internet. 32% of the ASes are found to have at least one critical link. Failures of these links, more prevalent than Tier-1 depeering, disrupt each affected AS's connections to more than 70% of the rest of ASes.

- The interdomain policy restricts the connectivity. 6% (hundreds) of ASes that are vulnerable to a critical link failure, in fact, have sufficient physical redundancy nearby to circumvent the failure if the imposed policy can be relaxed.

4.2.2 Case Study: Taiwan Earthquake

In today's Internet, failures occur quite frequently and sometimes with catastrophic consequences, *e.g.,* outages lasting for days. Typical events that causes failures include accidental cable cuts [75, 17, 18], hardware malfunction, power outage [20], natural disaster [21], human (*e.g.,* misconfiguration [41], maintenance or policy changes [61]), or even terrorist attacks [19]. Unfortunately, forecasting and statistically characterizing the occurrence of failures is too challenging to be accounted for in network design.

In Section 3.3.1, we did a case study on the Taiwan Earthquake in 2006. One of the key discoveries from the study is shown in Figure 4.1. We found that from the PlanetLab node in Japan

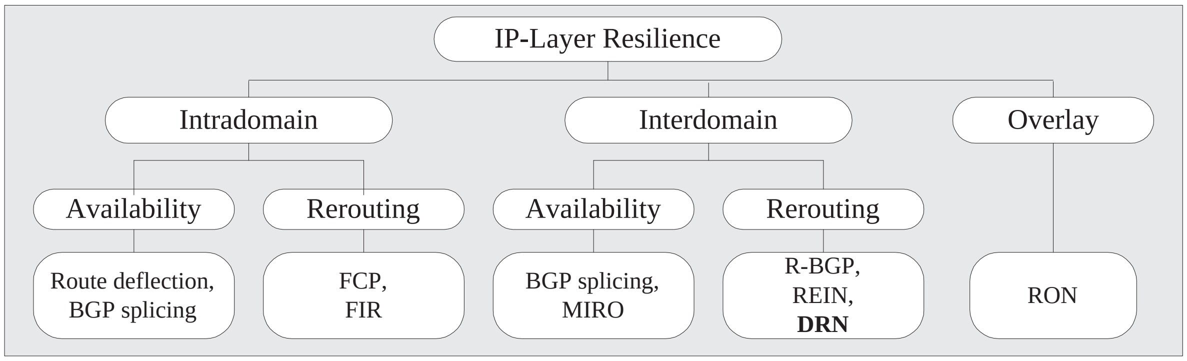

Figure 4.2: Taxonomy of techniques to improve resilience

to a Chinese commercial network, the path goes through the US, taking a long time to travel over an excessive physical distance. However, two networks in South Korea have direct connections to both Japan and Chinese networks. Hence, if the networks in Korea can provide temporary transit services for both China and Japan, we obtain an overlay path through Korea with a much shorter physical distance.

Bridging regional ISPs, despite possible violation of interdomain policies, can not only enhance the performance but also provide more resilience to failures. For example, regional ISP A and B establish a peer-peer relationship to benefit traffic between their customers. Meanwhile, A and B subscribe to different upstream providers. Suppose a failure disables the link between A and its only provider, A suffers outage until the link is repaired. Alternatively, A could ask B to provide a temporary transit service for A's traffic not only to B's customers but B's peers and providers (*i.e.*, all of the ASes B can access). Although emergency transit service might induce overhead (*e.g.*, B asks A for financial compensation), it would be a better choice for victims of recent oceanic undersea cable incidents [21, 75, 17, 18] than suffering from outages for days or even weeks.

4.2.3 Taxonomy of Techniques to Improve Routing Resilience

Traditional Internet routing protocols, *e.g.*, OSPF, BGP [80], dynamically react to network component failures by exchanging routing update messages to locate new viable paths. Until the routing convergence is completed, however, the network might undergo delays or even packet losses. Various techniques have been proposed to enhance the network resilience, each of which more or less tackles the problem from one of the following two angles.

- **Availability**: In today's Internet routing, packets are sent in only one direction, *i.e.,* to the next hop computed based on the shortest-path routing algorithm. Any fault along the path disrupts the flow of the packets. Techniques in this category propose ways to allow packets to be sent in multiple directions so that traffic flows cannot be easily broken up by a single fault. That is, each forwarding unit has high path availability.

- **Re-routing**: A disrupted traffic flow has to be re-routed via a new path to circumvent the failure. Techniques in this category aim to address how to expedite the computation of the new route or how to re-route without disrupting original traffic flows.

Figure 4.2 illustrates the taxonomy of the techniques that improve resilience in the traditional IP layer. Based on the applicable context, we first classify them into three categories: *intradomain*, *interdomain*, and *overlay*. We further distinguish intradomain and interdomain techniques by examining whether they provide high path availability or fast and undisruptive re-routing.

- **Intradomain**: In route deflection [76], routers forward packets to neighbors beyond the only next hop on the shortest path. BGP splicing [77] proposes to install multiple routes into the forwarding table at the ingress and egress routers to exploit the diversity of intradomain paths as well as interdomain peerings. On the other hand, FCP (Failure-Carrying Packets) [81] allows packets to automatically discover a working path without invoking the routing convergence. In FIR (Failure Insensitive Routing) [82], routers prepare for failures using interface-specific forwarding and trigger fast local rerouting using a data structure called "backwarding" table in the face of failures.

- **Interdomain**: BGP splicing described earlier also provides interdomain path diversity by allowing egress to choose multiple peering neighbors. In MIRO [78], each AS is allowed to negotiate with other ASes on the set of routes exchanged to attain more flexibility in path selection. MIRO can potentially provide ASes with high path availability to improve the network resilience. To achieve fast re-routing, R-BGP [79] precomputes a backup path for each prefix that is most disjoint from the primary path to eliminate the routing convergence. REIN [83] is proposed to allow neighbor ASes to arrange special interdomain paths to handle partitions inside one of the ASes.

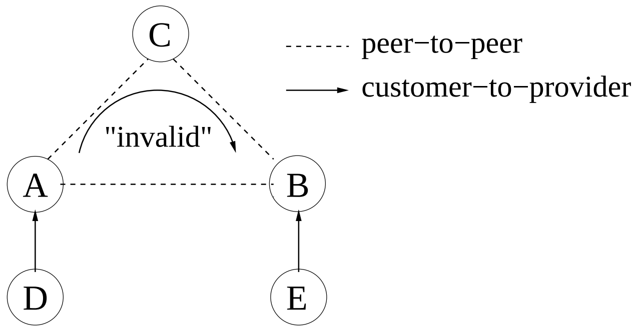

Figure 4.3: Example of BGP policy

- **Overlay**: RON [84] constructs a separate network among the set of overlay nodes on top of the underlying Internet infrastructure. By constantly maintaining the up-to-date overlay structure, the overlay routing is shown to provide abundant path diversity and resilience to failures in the underlying network.

In all of the techniques described above except MIRO and the overlay routing, routes that bypass the failed network components are subject to the current interdomain policy. Thus, they fail to tackle situations in which the policy restricts the selection of re-routing paths despite the existence of abundant physical redundancy. In this work, we would like to fill this void by developing a technique which can specifically address situations that other techniques cannot deal with. Our proposed approach can coexist with, and be complementary to, other existing approaches.

4.3 Dynamic Routing Negotiation

The dominant interdomain routing policy on today's Internet imposes "valley-free" restrictions: valid AS paths start with a sequence of consecutive customer-provider links, followed by zero or one peer-peer link, ending with a sequence of consecutive provider-customer links. Any subsequence of this path is also valid. Figure 4.3 illustrates an example of the effect of this restriction. A, B, and C have peer-peer links amongst them, while D and E connect through a customer-provider link to reach A and B, respectively. Node D uses path *[D A B E]* to reach node E, traversing a customer-provider link, followed by a peer-peer link, and reaching the destination through a provider-customer link. If the link between A and B breaks, D is disconnected from E, as D cannot use the path *[D A C B E]* due to policy restrictions.

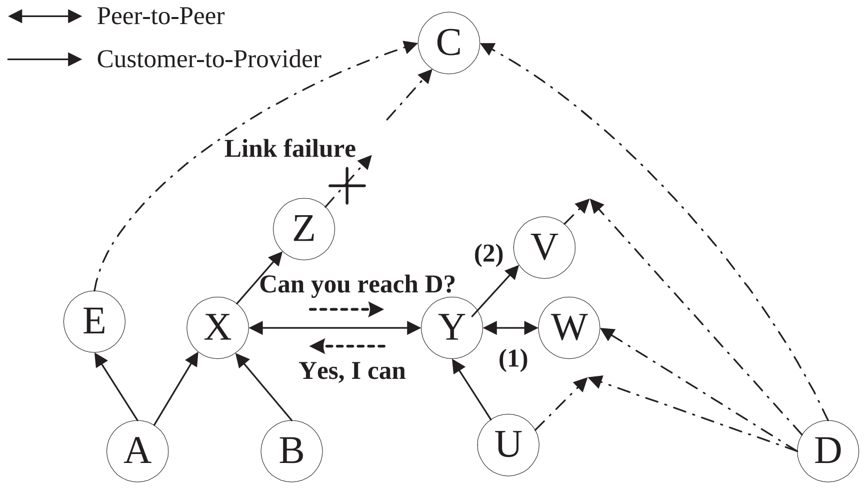

Figure 4.4: Achieving reachability by relaxing BGP policies

Policies are usually in place to enforce commercial relationships and also reflect the allocation of network resources to satisfy common traffic demands. We argue that Internet routing should be more *adaptive* with built-in mechanisms to dynamically relax routing policies for handling short-lived transit requests, particularly to neutralize the impact of severe failures, such as the recent Taiwan earthquake [21] or the 911 terrorist attacks event [19].

In what follows, we present a mechanism, called *Dynamic Routing Negotiation* (DRN), under which ASes are allowed to negotiate whether or not the original interdomain policy can be violated to provide more flexibility in selecting paths and utilize the redundant physical connectivity in the vicinity of a failed component. In Figure 4.3, if C relaxes its policy and provides A with a temporary transit service (*i.e.*, the link between A and C becomes equivalent to a customer-provider link), then D can reach E via path *[D A C B E]*. Next we detail the proposed mechanism and highlight key design features of DRN. We show that a routing policy can be easily relaxed by adjusting routing configurations on-the-fly without inducing much overhead.

4.3.1 The Proposed Approach, DRN

We use a simple example to illustrate the failure-recovery procedure by which policy-relaxed paths are identified through negotiations between neighboring ASes. In Figure 4.4, suppose a failure on the path between Z and C disrupts the *only* path from X to D. To verify whether or not the failure causes any *persistent* reachability problem, X first waits for a few minutes, the amount of time a typical interdomain routing convergence requires [59]. In this example, X eventually finds out that no viable path is available to bypass the failure, and therefore must seek alternative solutions. In DRN, X initiates contact with its neighboring ASes to see if any of them would and could provide reachability to D. Based on the relationship, X has three options: it can seek help from one of its providers, peers or customers. Obviously, all of X's providers (*e.g.*, Z) cannot reach D; otherwise, X could still reach D. Between the remaining two options, we argue that X prefers peers to customers because (1) peer-peer links usually have more bandwidth than provider-customer links; (2) peers are equipped with more internal resources than customers to accommodate the new traffic flow. In Section 4.4, we compare the performance enhancement between the negotiation with only peers and that with both peers and customers. For ease of exposition, we assume negotiation with peers in the rest of discussion.

So, X checks neighbors (*e.g.*, Y) that it has a peering relationship with. X first sends a request to neighbor Y, asking if it can forward X's traffic to D. Suppose Y has a valid path to reach D, then it has to go through either Y's peer via *path (1)* in the figure, or Y's provider via *path (2)*. If Y has sufficient resource to relay X's traffic to D, then it will reply to X with the possible financial cost incurred by this special service arrangement. If X agrees to the terms in Y's reply, Y will send X its best route to D upon completion of the negotiation. X can propagate this information further on the newly-learned route to its customers, *e.g.*, A and B. In this case, X and Y no longer follow the restrictions imposed by their normal peering agreements, and the new path *[X Y W D]* (suppose *path (1)* is taken by Y as its best path to D) is not "valley-free." In such a case, Y instead acts as a partial transit provider of X for it to reach destination D.

In DRN, each AS has its own process of speaking to its neighbors, which is independent of other ASes' actions. If the AS is unsuccessful in locating a new path after negotiations with its neighbors, it still could learn viable paths from its providers. In Figure 4.4, B re-connects to D via X's negotiated path.

4.3.2 Extending Negotiation beyond Immediate Neighbors

In the above example, an AS simply initiates a route negotiation with its immediate peering ASes or customer ASes. The negotiation can be extended to farther-away ASes than just immediate neighbors, if they are reachable even after the failure (*e.g.,* AS U in Figure 4.4). When the negotiation is extended beyond immediate neighbors, special care needs to be taken for proper data forwarding. Suppose X chooses U's path after a negotiation to reach D. When X's packet destined for D arrives at Y, it gets dropped because neither path *[X Y W D]* nor *[X Y V D]* is valid (Y still enforces the "valley-free" rule). The packet never reaches U. To avoid the intermediate ASes' tampering of the flow of packets, the two negotiating ASes (e.g., X and U) establish a tunnel to deliver the packets along the negotiated path. The initiator X assigns a local unique tunnel identifier during the negotiation and U maintains a tunnel id table in which each entry is associated with a tuple <AS, tunnel id> to uniquely identify the tunnel. X then directs all packets destined for D into the tunnel which are then extracted by U. After removing the header associated with the tunnel, U forwards the packets in a usual way until they reach D.

The AS initiating a negotiation often may have other requirements than just reachability. For example, the negotiated path must have sufficient bandwidth, or the cost must be within some acceptable limit. Extending the negotiation beyond immediate neighbors provides the AS more valid paths, one of which can then be selected to meet the AS's other requirements.

In summary, DRN slightly changes the policy imposed on the AS path, allowing the routing system to better utilize the inherent physical redundancy to enhance network resilience to failures. Most of policy restrictions are still retained in the selected AS path: the path between the requesting AS and the responding AS and that between the responding AS and the destination are both valid policy-restricted paths.

4.3.3 Advertising Negotiated BGP Routes

Next we discuss the advertisement of the negotiated BGP routes. In Figure 4.4, once X completes the negotiation with Y for destination D, Y sends X its best route to D. X then propagates the route further to its customers (*e.g.,* B) as if the route were learned from X's provider. Note that the basic route advertisement is uni-directional: Y does not advertise X to its non-customer neighboring ASes. So, Y only relays X's traffic to D. In this example, if Y propagates the route to X as if X were one of its customers, Y might attract traffic to X that used to traverse other paths (*e.g.,* via Z) and the amount of increased incoming traffic to X via Y is difficult to predict. In our basic mechanism, to reach X, D has to initiate its own negotiation process to locate an alternative path.

Techniques that attempt to control the flow of incoming traffic include AS-path prepending, selective prefix announcement, advertisement of more specific prefixes, and use of BGP communities. The control in the first three schemes are often coarse-grained. We discuss how to use BGP communities to set up a special arrangement so that only inbound traffic from D pass through Y in case bi-directional traffic control is necessary. To ensure that only X's inbound traffic from D traverses Y, Y advertises X's route in a way that only D learns of it. To achieve that, Y associated X's route with a BGP community BGP_NEGOTIATE with D as the community value. Y only exchanges this route information with the next-hop AS toward D, say W. Once W recognizes that the route carries BGP_NEGOTIATE community with D, W decides that this route can be exchanged with only the next-hop AS that it uses to reach D. The process continues until the route advertisement reaches D. This way, only the bi-directional traffic flows between X and D are permitted via Y. The drawback of using a BGP community is that it requires the cooperation of all ASes along the path.

We briefly discuss the address granularity at which the dynamic negotiation is performed. In Figure 4.4, each negotiation is done on a per-destination-prefix basis. Considering the large number of prefixes in the Internet, the negotiation process could incur significant overhead. To address this issue, each negotiation instead deals with multiple destination prefixes. For example, prefixes with the same AS origin certainly can be treated together. BGP atoms [85], which is defined as a set of prefixes that share the same AS path seen by BGP neighbors, can be used to further reduce the number of negotiation instances.

4.3.4 Route Convergence

Due to the expressiveness of the routing policy and the ASes' freedom in specifying their own policies, interdomain routing does not always converge. Fortunately, previous work by Gao *et al.* [86] has shown that the routing converges if ASes select and export routes based on conventional business relationships. Because DRN relaxes certain policy restrictions and the selected AS path no longer follows a business relationship, it could lead to routing divergence. Although route oscillations always can be dynamically detected and resolved by utilizing techniques proposed by [87], we discuss below how DRN imposes certain rules to inherently ensures the route convergence.

In DRN, a failure triggers a number of independent routing negotiations initiated by ASes that lost reachability to a given destination. For any given AS, it could have the following three types of routes: traditional policy-restricted route, its own negotiated policy-relaxed route, and negotiated routes learned from its provider ASes. We impose several rules in route selection.

- **Rule 1:** A conventional policy-restricted route is always preferred to the policy-relaxed route obtained via dynamic route negotiation.
- **Rule 2:** A policy-relaxed route is propagated *only* to customer ASes, as described in Section 4.3.3.
- **Rule 3:** Among policy-relaxed routes, different types of preference rules can been adopted. For example, the AS can choose as the best route the one with minimum AS hops, or the AS can choose the best route based on where each negotiated route is learned from. In case the AS always prefers routes from a specific provider AS, it can choose the negotiated route learned from that provider AS. As we will show later, the decision among policy-relaxed routes does not affect the routing convergence.

Next, we prove that under DRN, the routing still converges. Griffin *et al.* [4] showed that any route divergence or oscillation can be characterized by a *dispute wheel* in the network. In DRN, Rule 1 dictates that dynamically negotiated routes do not interfere with the traditional interdomain route propagation. Thus, we prove below that the propagation of the policy-relaxed routes does not lead to route divergence. By using proof-by-contradiction, we begin with an assumption that a dispute wheel exists. For a given destination d, nodes p_0, $p_1, \ldots, p_{n-1}$ are pivot nodes on the wheel. According to the definition of a dispute wheel in [4], for each pivot p_i, there exists a rim path to the next pivot $p_{(i+1)\ mod\ n}$. As dictated by Rule 2, the negotiated route is only propagated from the provider AS to the customer AS. So, p_i has to be p_{i+1}'s customer. Now, the wheel can be translated into an AS relationship loop. Such a relationship loop does not exist in the Internet; otherwise, coupled with the common practice of preferring customers' routes over peers' and providers' routes, the relationship loop essentially becomes a routing loop. Therefore, by contradiction, no dispute wheel exists and the routing still converges under DRN. Note that, Rule 2

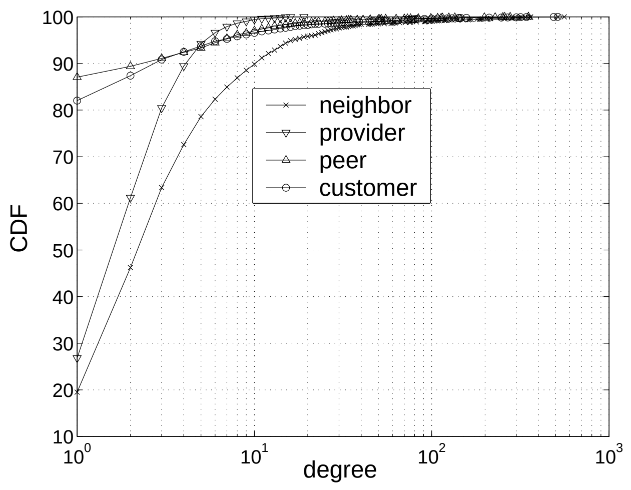

Figure 4.5: CDF of AS degree based on relationships

essentially ensures the route convergence and the preference rule among negotiated paths in Rule 3 does not interfere with the convergence problem.

4.3.5 Negotiation with Multiple Neighbors

In DRN, the AS initiating the negotiation prioritizes its neighboring ASes based on how far away they are from it. Requests are first sent to its immediate neighbors, then to neighbors two AS hops away, . . . , until all of the neighbors are contacted. In particular, among immediate neighbors, peers are preferred over customers. Not every negotiation is guaranteed to succeed because (1) the neighboring AS might have been affected by the same failure and lost reachability to the destination as well; (2) the neighbor AS might not have sufficient resources to accommodate the request. In practice, when an AS decides to initiate the search for a non-policy-compliant path to a destination, it often sends out negotiation requests to more than one of its neighbors simultaneously. Each negotiation, however, induces computation and communication overheads. Messages have to be exchanged and necessary computation needs to be done to decide whether non-policy-compliant paths can be established. Thus, each AS has to determine an optimal number of concurrent requests that trades off computation and communication overheads for increased likelihood of a successful negotiation.

Figure 4.5 presents the CDF of the AS degree (*i.e.*, the number of immediate neighbors) in a realistic Internet topology constructed based on BGP data collected from at RouteViews [38] and RIPE [43]. In our analysis, we choose 20 as the maximum number of neighbors that are negotiated simultaneously in a negotiation round. As shown in the figure, over 95% of the ASes have degree no more than 20. Thus, most of the ASes need only one negotiation round to contact all of their immediate neighbors. If the previous round does not result in the location of a non-policy-compliant path, a new round is invoked as the AS sends requests to the next 20 preferred neighbors. The process continues until one neighbor agrees to accommodate the request or all of the neighbors are contacted.

4.3.6 Neighbor Selection for Negotiation

As described earlier, the AS sends requests to its neighboring ASes for negotiation in the order that is based on its distance to them. Among neighbors with equal distance away, the AS randomly chooses the order of requests. Such selection based on distance and randomness is simple and easy to implement, however, it has drawbacks. For example, adjacent neighbors are likely to be impacted by the same failure. Sending requests to them thus yields no solution.

We propose a scheme to selectively choose neighbors that are unlikely to be affected by the same failure so as to perform a more efficient negotiation process. To achieve this, the AS first needs to obtain the approximate knowledge of the failure location so that it can determine more accurately if a neighbor still retains its reachability to the same destination. In our scheme, the AS constructs an AS-level topology graph based on BGP data in its own routing table or collected from public data repositories, such as RouteViews and RIPE. Second, by analyzing path changes for a set of destination prefixes, the AS can roughly identify the failure location (or where the routing change took place), which is similar to a BGP root-cause analysis [11].

Figure 4.6 presents a simple version of the root-cause analysis to estimate the possible faulty links or ASes. For each path change, the failed link has to be in the old path but not in the new path. If there is no path change, the failed link must not be in the path. Next, the AS uses the updated topology map with the set of links suspected to have failed removed in order to compute which of its neighbors still has paths to the destination. Figure 4.7 illustrates the complete procedure to select neighbors for negotiation based on the likelihood of circumventing the failure.

```
function locate_failure ($f$)
# $f$ is a failure, it returns $S$, the suspect set of failed links.
  $S$ = {};
  foreach prefix $p$
    if $p$'s route changes from $R_o$ to $R_n$
    # we consider failures, so $R_o$ is better than $R_n$;
      candidate_set $R_c$ = $R_o - R_n$;
      $S$ = $S \cup R_c$;
    elseif $p$'s route does not change
      $S$ = $S \cap R_o$;
  return $S$;
```

Figure 4.6: Pseudo-code to locate failures

```
function select_neighbor ($G$, $f$, $D$)
# $G$ is the topology graph, $f$ is a failure; $D$ is the destination
# it returns $N$, the set of neighbors for negotiation
   $N$ = {}; $S$ = locate_failure ($f$);
   $G'$ = $G - S$; # remove failed link from $G$;
  foreach neighbor $n$
    compute $n$'s shortest path to $D$;
    if $n$ is reachable to $D$
      $N = N \cup \{n\}$;
  return $N$;
```

Figure 4.7: Pseudo-code to select prospective neighbors

4.3.7 Reactive vs. Proactive

So far, we have considered the case in which each negotiation is triggered by failures. The disadvantage of this reactive approach is a long service disruption before a new path is found. To expedite the failure recovery, the AS can choose to perform negotiations prior to the occurrence of a failure. In case a failure occurs, it simply sends a message to the specific neighbor which already agrees on relaxing the conventional policy during the proactive negotiation and quickly activates the link for recovery.

The proactive scheme, however, has its limitations. First, the proactive scheme is performed prior to the occurrence of failures. For a given AS, to protect its reachability to a certain destination under any circumstance, the proactive negotiation has to find a non-policy-compliant path

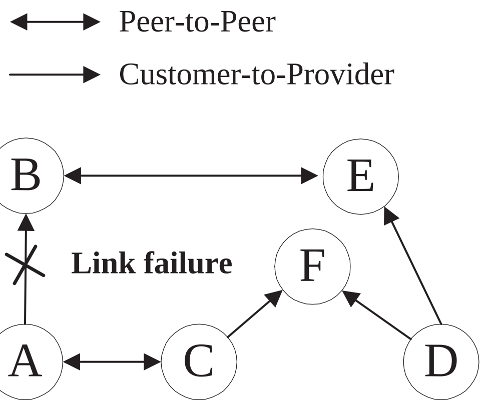

Figure 4.8: An example when R-BGP fails to recover.

that is completely disjoint from the current path. In contrast, the path obtained from the reactive scheme only needs to be free of the failed link(s) or AS(es). Obviously, the path selection is more restrictive in the proactive scheme. In Section 4.4.6, we extensively evaluate the limitation of the proactive scheme in selecting feasible paths. Second, in proactive scheme, resource might have to set aside in the neighbor to ensure that the path to be activated still can accommodate the negotiated requirement. In practice, reservation of resources in advance often incurs high financial cost.

4.3.8 Comparison with R-BGP

Recently, Kushman *et al.* [79] proposed R-BGP to improve network resilience by precomputing a backup AS path that is most disjoint from the primary path. If non-policy-compliant paths are allowed as backup paths, R-BGP is shown to reduce the number of disconnections resulting from a link failure down to zero. In Figure 4.8 we demonstrate that our proposed DRN can deal with certain failures that R-BGP cannot. In R-BGP, the backup path is only advertised from an AS to its neighbor it uses as the next-hop AS to reach the destination. In the example, A reaches D via B, and C reaches D through one of its providers F. C does not advertise any backup path to A because C currently does not use A as the next hop to D. So, A cannot learn C's path as a potential backup even if R-BGP is allowed to compute a policy-non-compliant path. If the link between A and B fails, A will experience loss of reachability to D. DRN, however, can deal with this failure once A detects loss of reachability to D. This simple example illustrates the key benefit of DRN.

4.3.9 Practical Considerations

DRN assumes that ISPs are willing to provide temporary transit services to each other to overcome non-transient failures. Clearly, this requires cooperation and incentives for wide-spread deployment. We leave DRN's pricing design as future work. Even limited deployment can significantly enhance the resilience of Internet routing, and DRN can be incrementally deployed. We argue that some form of cooperation to provide partial transit under emergencies already exist as described by the REIN protocol [83] addressing a special case of failures, network partitions, but mostly it is coordinated manually. Our proposed design automates this, improving the efficiency and effectiveness of the coordination.

Note that negotiated BGP routes will not introduce any prolonged routing convergence as they are negotiated between pairs of ASes. The newly-negotiated routes are only propagated downstream to one's customers. To provide bidirectional transit, route advertisement is restricted along an existing path without causing any increase of delay in the existing routing convergence process. Our design is also scalable with a manageable increase in router state through aggregation at the BGP atom level.

As discussed in Section 4.3.1, DRN is triggered when the AS discovers that it no longer has a valid route to the destination for a period of time each routing convergence typically lasts. Sometimes routing convergence takes longer and final stable route might arrive after DRN is activated. As discussed in Section 4.3.4, the conventional policy-compliant route is always preferred over the non-policy-compliant route, thus the AS immediately terminates the DRN and discards any of the routes learned from DRN.

4.4 Evaluation

In this section, we evaluate DRN via extensive simulation on realistic Internet topologies. First, we present the simulation setup and describe the types of failure scenarios as well as the metrics we use in evaluation. Then, we discuss the simulation results for two major types of failures that lead to unreachability. We also briefly discuss simultaneous failure of multiple links. Finally, we evaluate the limitation of performing DRN proactively.

4.4.1 Simulation Setup

We evaluate the performance of DRN on an AS-level network topology. To construct realistic Internet topologies, we use two months of BGP data in the form of routing table snapshots as well as routing updates from RouteViews [38], RIPE [43], public route servers [44] as well as a large content distribution network from March to April 2007. The measurement data were collected from vantage points located in a total of 483 different ASes. To reduce the size of the network graph and speed up our analysis, we prune the graph by eliminating *stub* nodes, *i.e.,* customer ASes that do not provide transit service to any other AS. These can be easily identified from routing data as ASes that appear only as the last-hop ASes but never as intermediate ASes in the AS paths. As a result, we could eliminate 63% of the links and 83% of the nodes. During the analysis on the enhancement achieved by DRN, we restore such information by tracking at each AS node in the remaining graph the number of stub customer nodes it connects to, including information regarding whether they are single-homed or multi-homed to other ISPs.

Next, we label each link in the topology graph with one of three basic AS relationships—customer-provider, peer-peer, and sibling—to infer valid, policy-conforming AS paths [52]. Despite the recent efforts on inferring AS relationships [51, 45, 49, 50, 53], constructing a topology graph that matches exactly the current Internet is impossible due to the lack of knowledge of the proprietary relationship information. We attempt to create a topology with best accuracy for our analysis. A recent study [54] shows that the latest Gao's algorithm [51, 50] and CAIDA algorithm [49] present better accuracy in satisfying the traditional "valley-free" [51] policy rule for most AS paths. So, we first generate a graph using Gao's algorithm with a set of 9 well-known Tier-1 ASes (AS 174, 209, 701, 1239, 2914, 3356, 3549, 3561, 7018) as its initial input. Then, we compare the computed graph with graph CAIDA from [48]. We take the set of AS relationships agreed on by both graphs, which we believe are most likely correct, as the new initial input to re-run Gao's algorithm to produce the graph for our analysis. Table 4.1 presents the basic statistics of the constructed topology graph. We admit that the constructed topology does not exactly match the real Internet. The BGP data collected from a limited number of vantage points cannot locate all of the links [46]. Besides, the AS relationship, inferred based on heuristics, is not perfect. We address these issues by adding the low-tier peering links discovered by He *et al.* [47] and perturbing the relationship on a set of ASes to examine the effects on the simulation results. It is found that the

Property	Value
# of AS nodes	4427
# of AS links	26070
# of customer-provider links	14343 (55.0%)
# of peer-peer links	11446 (43.9%)
# of sibling links	281 (1.1%)

Table 4.1: Basic statistics of constructed topology

topology inaccuracy does not alter the fundamental conclusion. Due to the limited space, we only present the simulation results on the basic topology.

As described in Section 4.3, DRN is intended to improve the Internet routing resilience when the interdomain policy restricts the selection of paths that could bypass the failures. Thus, our evaluation focuses on the failure scenarios in which unreachability is caused by policy restrictions, not by insufficient physical redundancy. According to our previous study, the interdomain reachability disruption in today's Internet is caused by two types of failures: *Tier-1 depeering* and *failures of critical customer-provider links*. We present the simulation results for these two types of failures in Sections 4.4.2 and 4.4.3, respectively.

In each simulation test, we compare the current BGP with variants of DRN based on the *scope* of search. Here scope is the *maximum* AS path length allowed between the pair of ASes in negotiation. In particular, when scope is one (i.e., affected ASes negotiate with their immediate neighbors), we consider the following two schemes.

- **DRN with peers**: When selecting neighbors to bypass a failure, peers are always prioritized because peering links usually have more bandwidth. Besides, peers usually have similar network size and comparable internal resources to carry the diverted traffic. In this scheme, negotiations are only performed between peers.

- **DRN with peers and customers**: The links to customers are less preferable because of their limited bandwidth. In practice, however, these access links might be just rate-limited and physically they can accommodate high-bandwidth requests. In this scheme, negotiation requests are sent to the peers as well as the customers of the affected AS(es).

In each instance of DRN, an AS, which loses reachability to a destination AS, sends requests to a set of ASes, attempting to locate policy-relaxed paths to re-gain the reachability. In the basic setup, the AS always prefers its immediate neighbors, then neighbors two AS-hops away, ..., until the search scope is reached. As described in Section 4.3.5, each negotiation process is divided into *rounds*, each of which can have a maximum of a predefined threshold number of ASes. In our simulation, we set this threshold to 20. Also, note that we currently do not consider the bandwidth requirement during the negotiation due to the lack of link-usage information.

To quantify the performance of DRN, we develop the following metrics:

- **The percentage of connected AS pairs, denoted by** R: In each test, a failure, depending on its type, disrupts communication between a vastly different number of pairs of ASes. In order to simplify the comparison between different routing schemes, we use the percentage of the connected AS pairs over the maximum number of AS pairs that could potentially fail to quantify the relative damages under different recovery schemes. For example, depeering between two Tier-1 ASes, A and B, disconnects A's single-homed customers from B's single-homed customers. The maximum number of AS pairs that could fail can be calculated by multiplying the number of A's single-homed customers by the number of B's single-homed customers.

- **The number** N **of neighbors negotiated**: This metric represents the number of neighbors negotiated to restore the reachability between a pair of ASes. N can be regarded as an indicator of the overhead induced by DRN. The larger N, the more computation and communication overheads it incurs.

- **The percentage** V **of successful negotiations**: Not every AS negotiated has a valid path to the destination as it too might be affected by the same failure. V can be viewed as an indicator of the difficulty in locating a valid path via DRN. In practice, the larger V, the more choices there are for selecting an AS that might also satisfy other requirements like bandwidth and financial costs.

- **The number** T **of rounds taken before locating a viable path**: T can be viewed as an indicator of the time spent in each negotiation before locating a new path in practice.

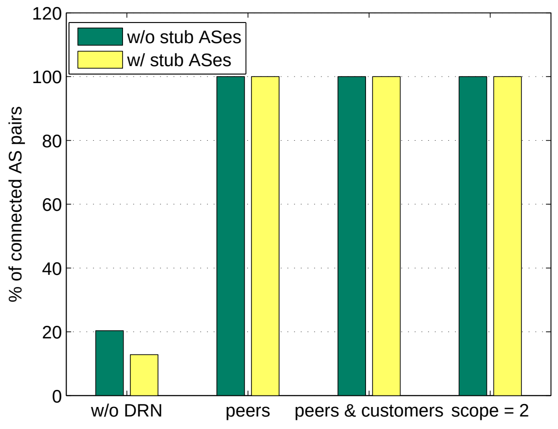

Figure 4.9: Connected AS pairs in Tier-1 depeering

Scheme	N_{avg}	$V_{avg}(\%)$	T_{avg}	S_{avg}
peers (scope=1)	27.3 (7.7)	100.0 (0.0)	1.0	1.22
peers & customers (scope=1)	224.5 (33.5)	92.3 (1.7)	1.0	1.19
scope=2	1537.6 (165.1)	97.4 (1.1)	1.0	1.19
scope=3	3178.2 (207.2)	99.0 (0.6)	1.0	1.18
scope=4	3951.3 (243.6)	99.3 (0.3)	1.0	1.18

Table 4.2: Performance of DRN in Tier-1 depeering

- **The stretch** S: can be calculated by dividing the length of the AS-level path located by DRN by the length of the old path, indicating the "path inflation" by DRN.

4.4.2 Tier-1 Depeering

Our first analysis evaluates how DRN improves the network resilience under Tier-1 depeering. Today's Internet core consists of a group of large ISPs known as *Tier-1 ASes* which are the top service providers. Their customers reach each other via the peering links among the Tier-1 ASes, so these peering links are of utmost importance to maintaining the Internet connectivity. So, Tier-1 depeering could cause severe damage to the Internet. In particular, single-homed customers of the two affected Tier-1 ASes usually cannot reach each other.

We ran a total of 36 Tier-1 depeering tests in each of which a peering relationship among the 9

Tier-1 ASes is broken. Figure 4.9 shows the percentage R of connected AS pairs under traditional BGP, DRN with peers, DRN with both peers and customers, and DRN with scope 2. Suppose C_i and C_j are the number of single-homed ASes for Tier-1 AS i and AS j that are depeered. R can be calculated as

$$R_{i,j} = \frac{\#\, of\ connected\ AS\ pairs}{1/2 \times C_i \times C_j}$$

.

Note that for simplicity our network topology used in the analysis does not contain any stub ASes. By tracking the types of links connecting these stub ASes, we also calculate the results when the stub ASes are restored. Under the traditional BGP, single-homed customer ASes of the depeered Tier-1 ASes have to rely on the low-tier peering to reconnect. As shown in the figure, only about 20% of these AS pairs are able to maintain their communication by low-tier connectivity. DRN with only peers, however, can restore all of the remaining 80% of the disconnected AS pairs by allowing certain peer-to-peer link to provide temporary transit service. During the depeering, in addition to the single-homed customer ASes, the two Tier-1 ASes themselves can no longer reach each other. Either Tier-1 AS starts the negotiation by sending requests to its peers, which includes other Tier-1 ASes. Because of the abundant connectivity associated with these Tier-1 ASes, it can quickly identify a peer AS which can reach the disconnected ASes. The newly-negotiated route is then further propagated to its customers, thus re-gaining their reachability. In Tier-1 depeering, negotiation with only peers is sufficient to restore communication between all of the disconnected AS pairs and negotiation with customers or a larger search scope does not make any improvement.

Table 4.2 presents the results of the other metrics for variants of DRN. The value in a parenthesis is the standard deviation for the corresponding metric. As shown in the table, with the increase of the scope, each AS is able to request a rapidly-increasing number of ASes to establish policy-relaxed paths. A high value of V_{avg} indicates that, under depeering, the affected ASes can locate many feasible recovery paths with DRN. The recovery path can be identified within 1 round and does not exhibit any significant path inflation.

```
function find_path($src$, $dst$, $last$, $link\_set$)
# if returns TRUE, paths exist between $src$ and $dst$;
# $link\_set$ is the set of links shared by these paths
  if ($src$ = $dst$)
    ret = TRUE; $link\_set$ = $\{(last, dst)\}$
  else
    $S$ = $\{$all links$\}$; ret = FALSE; # initialize S and ret
    foreach $x \in \{src$'s providers or siblings$\}$
      if (find_path($x$, $dst$, $src$, $S_x$) = TRUE)
        $S = S \cap S_x$; ret = TRUE;
    $link\_set = S \cup \{(last, src)\}$;
  return ret;
```

Figure 4.10: Algorithm to locate shared links among all paths from src to dst.

4.4.3 Failures of Customer-Provider Links

Most AS links are either peer-peer or customer-provider links. For peer-peer links, only Tier-1 depeering, which is evaluated in Section 4.4.2, causes unreachability. Here we evaluate how DRN performs in the face of customer-provider link failures. First, we describe the methodology for identifying the set of failures that are of most interest to us. Then, we present the simulation results for these failures.

DRN aims to improve Internet routing resilience by relaxing policy restriction and better utilizing the existing physical redundancy. Therefore, we select only the failure which results in unreachability without physically partitioning the topology. Given that Tier-1 ISPs are richly connected, the robustness of connectivity of an AS can be captured by the similarity of its paths reaching the Tier-1 ASes. *Path similarity* can be defined as the number of commonly-shared links among all the paths under consideration. In particular, a nonzero path similarity means that failing a *single* link can disrupt reachability. We develop a recursive algorithm for finding the set of all commonly-shared links among all possible paths between a given non-Tier-1 AS and the set of Tier-1 ASes, shown in Figure 4.10. By applying the algorithm to the topology, we found that 958 of 4418 non-Tier-1 ASes share common customer-provider link(s) to reach Tier-1 ASes.

Not all of the 958 instances, however, satisfy our requirement because breaking some of the commonly-shared links might physically partition the graph. To identify the set of ASes that can be physically disconnected from the network by a link failure, we remove all the relationship labels

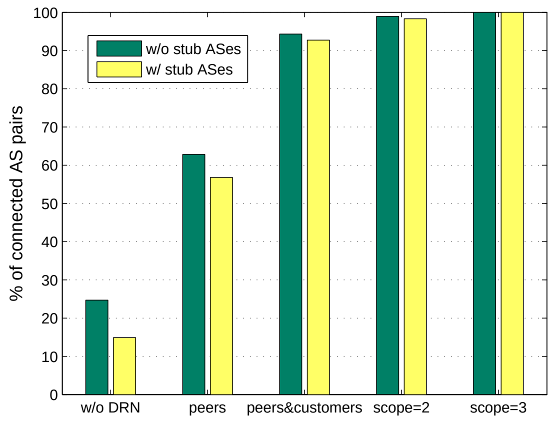

Figure 4.11: Connected AS pairs under failures of critical customer-provider links

on the links and convert the topology graph into an undirected graph. The problem of locating physically critical link reduces to a max-flow-min-cut problem [66]. 703 out of 4418 non-Tier-1 ASes are found to have a min-cut value of 1 and breaking one of the shared links disconnects the corresponding AS from the rest of the network. No physical redundancy has been provided for these ASes. By eliminating 703 ASes from our initial set of 958 ASes, we have 255 ASes that are susceptible to a single customer-provider link failure despite the presence of adjacent physical connectivity.

We identify a total of 255 customer-provider links and in each simulation test, we remove one of these links and evaluate how different routing schemes react to the failures. Note that each critical link failure might disconnect a set of ASes from the rest of the Internet. The metric R_l of critical link l is calculated as follows.

$$R_l = \frac{\#\ of\ connected\ AS\ pairs}{1/2 \times C_l \times (C - C_l)}$$

,

where C_l is the number of ASes that must traverse l to reach Tier-1 ASes and C the total number of ASes in the graph. we calculate the results for the cases without stub ASes as well as with stub ASes. Figure 4.11 illustrates the percentage of connected AS pairs under five different schemes. Under the traditional BGP, when an AS loses its connection to Tier-1 ASes, it loses reachability to an average of 75% of ASes. It maintains connection to other ASes only via low-tier peering links

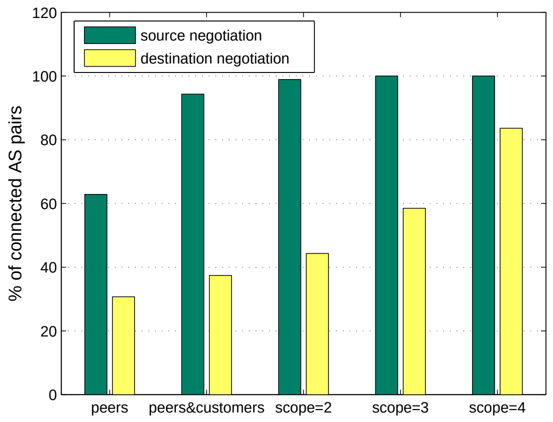

Figure 4.12: Connected AS pairs in two directions

or via low-tier common providers. DRN with peers reconnects about 38% of AS pairs, reducing the percentage of unreachability down to 37%. In contrast to results in Section 4.4.2, negotiation with only peers does not guarantee the full recovery from these types of failures. It is due mainly to the limited connectivity of the affected ASes, for example, some ASes simply do not have peers at all. If customers are allowed to negotiation, DRN can further reduce the unreachability down to 1.9%, i.e., almost reconnecting all of the disrupted AS pairs. The remaining unreachability can be restored by increasing the search scope to 3.

As described in Section 4.3, when a pair of ASes lose reachability to each other, either AS can start its own negotiation process to locate a new path independently. Unlike Tier-1 depeering in which the failed link is located in the middle of the path between the disconnected AS pairs, the failed link in case of customer-provider link failures is often *"unbalanced"*, meaning that the failed link is much closer to one of the ASes than the other. To investigate the difference in the performance with regard to the distance to the location of the failures, we examine DRN in both directions. Here, we refer to the AS which is closer to the failure, *i.e.*, the AS all of whose paths to reach Tier-1 ASes share the failed link, the *source*, and refer to the other AS as the *destination*. Figure 4.12 presents the percentage of connected AS pairs for both directions under five variants of DRN. As shown in the figure, the source recovery shows significantly better performance than the destination recovery. Even increasing the search scope to 4 cannot recover all of the unreachability from the destination. This can be explained by the types of paths that can be potentially located

Scheme	N_{avg}^{src}	V_{avg}^{src} (%)	T_{avg}^{src}	S_{avg}^{src}	N_{avg}^{dst}	V_{avg}^{dst} (%)	T_{avg}^{dst}	S_{avg}^{dst}
peers (scope = 1)	2.8 (0.7)	97.3 (2.1)	1.0	1.17	4.5 (2.8)	6.8 (4.7)	1.0	1.21
peers & customers (scope = 1)	6.3 (3.3)	94.7 (2.4)	1.0	1.14	11.1 (6.2)	5.4 (3.4)	1.0	1.18
scope = 2	13.0 (6.2)	96.3 (1.9)	1.0	1.13	57.4 (11.3)	8.7 (5.7)	1.9	1.16
scope = 3	18.5 (7.7)	98.1 (1.7)	1.0	1.13	235.7 (31.5)	11.5 (6.3)	2.4	1.14
scope = 4	25.1 (9.6)	98.9 (1.6)	1.0	1.13	518.4 (73.1)	6.3 (2.5)	2.7	1.13

Table 4.3: Performance of dynamic negotiation in customer-provider link failures

during the process. For recovery from the destination side, whichever peer or customer it selects has limited connectivity to the source because the source can only reach about 25% of the rest of the ASes after the failure in Figure 4.11. In contrast, recovery from the source allows the source to locate a neighbor which is not affected by the failure and retains the reachability to the rest of the Internet.

Table 4.3 shows the performance of the different schemes in terms of other metrics. We also separate the analysis into two directions. The results show the following characteristics. DRN from the source only sends requests to a small number of ASes (R_{avg}^{src}) while achieving a higher successful rate (V_{avg}^{src}) within a short period of time (T_{avg}^{src}). In contrast, DRN from the destination needs to contact a rapidly-increasing number of ASes with an extremely low successful rate, incurring 2 to 3 rounds of negotiations. In both schemes, however, DRN does not cause significant path inflation.

In summary, our simulation results in Section 4.4.2 and Section 4.4.3 show that DRN can quickly bypass the failure by seeking temporary transit services from the adjacent ASes without inducing too much overhead. The success of DRN in locating new non-policy-compliant paths demonstrates that the Internet has abundant physical connectivity, and DRN makes a significant improvement of the Internet resilience with only a small, short-lived digression from the traditional interdomain routing paradigm. Meanwhile, in case of critical customer-provider link failures, the performance of the negotiation scheme exhibits a significant difference as the recovery originated from a distant site to the failure achieves a worse success rate with more overhead.

Scheme	T_{avg}^{random}	T_{avg}^{local}
peers (scope=1)	1.0	1.0
peers & customers (scope=1)	1.0	1.0
scope=2	1.9	1.0
scope=3	2.4	1.0
scope=4	2.7	1.0

Table 4.4: Enhancement of local decision in recovering customer-provider link failures from the destinations

4.4.4 Selection of Neighbors

In Section 4.3.6, we propose that, when selecting neighbors to send requests, each AS can make a more intelligent decision based on the available local information and prioritize the selected neighbors according to the likelihood of having a viable path after a failure. Here we evaluate the improvement by making a local decision rather than selecting randomly.

We have shown that random selection works well under most of the situations considered in Section 4.4.2 and Section 4.4.3 because neighboring ASes rarely fail simultaneously and can thus be used during emergency. However, in case a distant customer-provider link failure disconnects an AS from the rest of the network, it becomes extremely difficult to locate a neighbor to reconnect to that AS from the destination side because most of the ASes have also lost their connectivity to it. An intelligent local decision to prioritize the neighbors to negotiate now becomes desirable despite the extra computation overhead.

Table 4.4 shows the improvement of decision based on local information in recovering customer-provider link failures from the destinations. Using decisions based on the local information can estimate for each negotiated AS the likelihood of having a viable path to bypass the failure upon which the negotiation requests are prioritized. As shown in the table, DRN with local decision always can locate a valid path within one round. In practice, however, the benefits of adopting an intelligent decision might not be as significant as the simulation results suggest because of the inaccuracy in inferring the root-cause of each routing change.

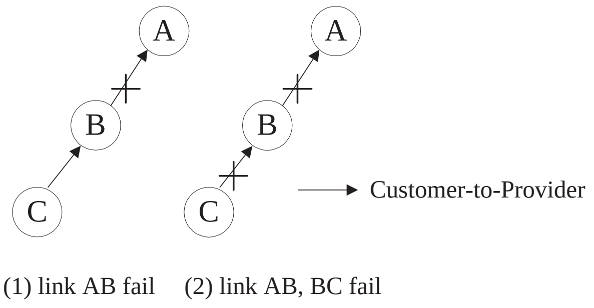

Figure 4.13: Multiple link failures

4.4.5 Multiple Link Failures

In our simulation, we also test how DRN performs in case of multiple simultaneous critical link failures. Especially, we consider the scenario exemplified by Figure 4.13. In the figure, link *[B C]* is a critical link of C and link *[A B]* is a critical link of both B and C. In each test, we compare how the routing reacts to the failure of link *[A B]* and to the failure of both links. For C, losing link *[B C]* disconnects it from more destinations which would otherwise be reachable via *[B C]*. For B, losing *[B C]* prevents B from contacting C for a route negotiation, thus diminishing the improvement of DRN.

We have a total of 18 test cases in each of which we examine how B and C react to the failure, and evaluate the effects of multiple link failures by comparing R before negotiation and $N * V$ (***i.e.***, the number of successful negotiations) relative to the corresponding single critical link failure, ***e.g.***,

$$R_{relative} = \frac{R_{AB\ fails} - R_{AB,\ BC\ fail}}{R_{AB\ fails}}$$

.

R before negotiation is used to capture the reachability impact of the failure and $N * V$ measures how well DRN locates the policy-relaxed paths. Our simulation shows that the averages of $R_{relative}$ and $N * V_{relative}$ are 27.3% and 37.6%, respectively. Multiple simultaneous critical link failures are shown to have a larger impact on the network reachability by disconnecting more pairs of ASes and reducing the number of alternative paths for recovery from failures under DRN.

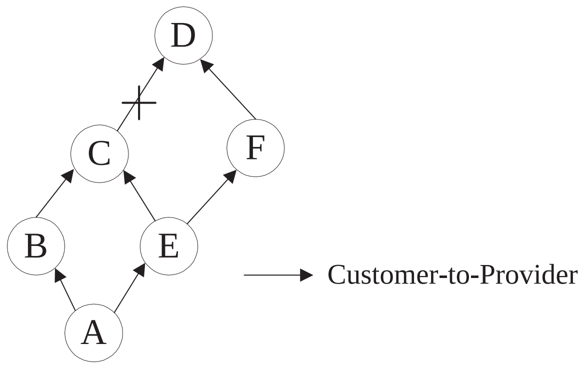

Figure 4.14: Limitation in proactive negotiation

4.4.6 Proactive Negotiation

Section 4.3.7 discusses the benefits of proactive negotiation: an AS no longer experiences a long delay before locating a new path. Instead, it quickly activates the link it already negotiated before a failure takes place. Since the proactive negotiation is performed prior to the occurrence of any failure in the current valid path, a successfully-negotiated path has to be disjoint from the current path to guarantee the restoration of reachability from any type of failure. In contrast, a reactive negotiation takes place only after a failure and thus, a negotiated path only needs to avoid the failed link(s). Because the links on the current path are less likely to fail simultaneously, the reactive approach tends to have more valid negotiated paths. Figure 4.14 illustrates the limitation of proactive negotiation. Suppose B attempts to locate a negotiated path prior to any failure in its path to D. B proactively initiates negotiation with A. Upon receipt of the negotiation request, A examines all its possible paths to D, which are paths *[A B C D]* and *[A E C D]* (suppose E currently chooses *[E C D]*). Both paths overlap with B's current path *[B C D]* and thus do not guarantee protection from failures on the path between B and D. In contrast, if the reactive approach is adopted, A has viable negotiated paths for any failure associated with B. Suppose link *[C D]* fails, E switches its path to *[E F D]* and then propagates it to A, which can be used by A to recover from the unreachability between B and D. A has no knowledge of path *[E F D]* prior to the failure because E only advertises the best path *[E C D]*. In practice, ASes could relax its requirement during a proactive negotiation in order to locate a backup path; for example, B might seek backup paths which avoid only one of its critical link *[C D]* instead of all links in the path. Still, as revealed in the example, B fails to locate any backup because some of viable paths might appear only after

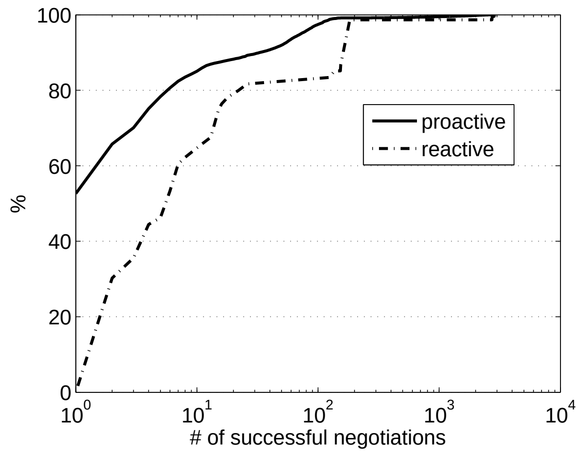

Figure 4.15: Proactive negotiation vs. reactive negotiation

a failure actually occurs.

In our simulation, we randomly generate 100,000 pairs of ASes in the topology and compare the number of possible successful negotiations in the proactive scheme with that in the reactive scheme. Since we are interested in those failures that cause unreachability, at least one of a pair of ASes is vulnerable to critical link failure discussed in Section 4.4.3. In the reactive scheme, we consider the worse case in which the corresponding critical link fails. Our analysis does not set a limit to the scope of negotiation and any AS in the topology can be used to bypass the failure. Figure 4.15 shows the cumulative distribution function of the number of successful negotiations for proactive and reactive schemes, respectively. To accommodate the logarithmic x-axis, we intentionally add 1 to the x value of each data point. As shown in the figure, more than 50% of pairs of ASes cannot locate a backup path with the proactive scheme while the corresponding AS pairs are able to locate at least one path to bypass the critical link failure. Overall, the average number of successful negotiations under the proactive and reactive schemes is 19.3 and 70.1, respectively. Despite the benefit of its quick reaction to failures, the proactive scheme is shown to have difficulty in locating feasible solutions, especially during severe failures when the degree of physical redundancy is low.

4.5 Related Work

Ensuring network survivability in the presence of failures has been studied in a variety of contexts, such as protection and restoration in WDM optical networks [88], virtual path routing in ATM networks [89], as well as the recent label-switching technique in MPLS networks [90].

In traditional IP-layer, various approaches have been proposed to improve the Internet resilience to failures in intradomain routing as well as interdomain routing, either providing high path availability [76, 77, 78] during packet forwarding or performing fast re-routing by reducing [91, 92, 93] or completely eliminating the routing convergence [79, 81, 82, 83]. In interdomain routing, Xu *et al.* [78] proposes a mechanism, called MIRO, that allows neighboring ASes to negotiate multiple BGP routes. MIRO can potentially enhance the network resilience as ASes acquire more paths to the destination. Despite the commonality of inter-AS route negotiation, our work differs from MIRO in that we provide a more thorough analysis on how to sacrifice a small degree of policy compliance for the sake of improved resilience by utilizing the inherent Internet physical redundancy under extreme failure conditions. To achieve fast re-routing, some prior work has focused on how to expedite the routing convergence and reduce the number of messages exchanged in the process. R-BGP [79] precomputes a backup AS path for each prefix that is most disjoint from the primary path. Under R-BGP, routing convergence no longer causes packet losses. DRN differs markedly from R-BGP and other prior work as DRN specifically addresses problems when no policy-compliant paths are available.

Inter-AS negotiation has also been utilized in a few studies with different objectives. Mahajan *et al.* [94, 95] proposed that ISPs be allowed to negotiate and jointly control the routing so as to achieve better end-to-end performance.

4.6 Concluding Remarks

In this chapter, we presented a new mechanism called *Dynamic Routing Negotiation* (DRN) that allows an AS to dynamically negotiate with its neighbor ASes for routes with relaxed routing policies to better utilize the physical redundancy in the network. DRN gives the interdomain routing more flexibility and adaptability in selecting the paths, *i.e.*, temporary non-policy-compliant paths, especially when it experiences severe reachability problems caused by failures. Our in-depth simulation analysis on realistic Internet topologies demonstrates that, when there is enough physical redundancy in the proximity of the failed location, DRN can reduce reachability loss during Tier-1 depeering and critical customer-provider link failures down to 0% and 37% by negotiating only with peers from 81% and 75%, respectively. Further resilience enhancement can be achieved if customers or neighbors farther away are negotiated.

CHAPTER 5

Uncovering Resource Sharing in MPLS Networks

The knowledge of network topology plays an important role in diagnosing network anomalies and devising measures to alleviate their effects. In this chapter, we go back to the reactive approach as in Chapter 2 and make important contributions in enhancing network reliability by improving the transparency of today's increasingly opaque networks.

5.1 Introduction

In today's Internet, each customer network pays to establish a transit service with its provider ISP to reach the rest of the Internet. Networks of comparable sizes and traffic demands establish peering links without cost to facilitate traffic exchanged between their respective customer networks. Typically, a SLA (Service Level Agreement) specifies the quality of the service each network expects in the treatment of its traffic traversing its neighboring networks. Once traffic enters its neighbor, the current network has no control over how its traffic traverses its neighbor's network. Performance requirements are usually met as the neighbor network strives to provide the promised quality of service. Unfortunately, when a certain network anomaly occurs, simply relying on its neighbor to respond might not be the best solution as it might take a long time before the anomaly is detected and then resolved.

Let us look at the ISP topology with five border routers in Figure 5.1. For R_1, its internal paths to R_3, R_4, and R_5 share a common segment between the black node and R_1. Suppose there is a failure on the commonly shared segment. Traffic that enters at R_1 and is destined for d experiences degraded performance. The customer network could choose to wait for the ISP to

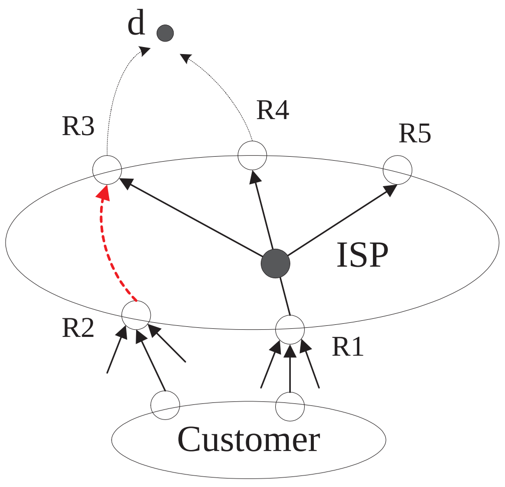

Figure 5.1: An example network topology to illustrate resource sharing.

resolve the failure and sustain a period of poor service. Instead, by knowing that R_2's internal path is not affected by the failure, it could also take a more proactive approach by diverting its traffic internally to enter the ISP at R_2. Having knowledge of the neighboring network's topology is thus essential to the adoption of such proactive measure to alleviate impacts of network anomalies. This is also applicable for other ISPs who peer with each other, especially given potentially numerous peering locations.

Traditional measurement tools, such as traceroute, can be used to acquire the individual hops between the probing source and a specific destination. The effectiveness of these active probing tools, however, is dictated by the responsiveness of the devices in the network. Such cooperation does not always exist in practice and might be less common as privacy concerns increase. In addition, the fear of the emerging malicious activities such as denial-of-service (DOS) attacks and Internet worms may further force network administrators to block response to these diagnostic tools (ping and traceroute). Lastly, the popularity of the Internet and the advance of technology has driven ISPs to integrate together a variety of different infrastructures. For example, the unification of Layer-2 (*e.g.*, MPLS) with Layer-3 IP service might make it more impossible to use traceroute as the path information could now be hidden in the lower layer. It has been confirmed by [96] that some ISP using a circuit technology such as MPLS is found to be "highly connected" and the actual path taken, which tunnel through PoPs, is not visible in the traceroute data.

It is thus important to be able to infer the network topology without requiring the cooperation of internal devices in the network. In this study, we only assume the availability of end-to-end measurements across each network, that is, the measurements from each ingress to each egress. The identification of topological information from end-to-end measurements has been studied extensively in network tomography [97, 98, 99, 100, 101, 102, 103]. These studies, however, mostly focus on inferring a tree structure connecting a single sender to multiple receivers. The identification of a more complete topology across multiple senders and receivers has not been fully addressed. In our study, we divide the inference of the network topology into two tasks. We first construct for each ingress a tree based on the end-to-end measurements from the ingress to other egresses. In the second task, we propose techniques to complement the tree construction procedure and merge different trees together to identify a more complete topology. With the limited knowledge extracted from the end-to-end measurements, the inferred network topology does not exactly match the network internal structure. However, the inferred topology captures to a certain degree how resources are shared across the ingress and egress nodes and can effectively assist us in pinpointing the origin of network anomalies and then suggest better proactive measures to alleviate the impact of performance problems.

Through extensive simulations, we demonstrate the feasibility of the proposed techniques. In particular, the tree merge algorithm correctly clusters receivers in more than 90% of the cases considered. Using both the loss rate and the delay metric, the clustering accuracy of the tree merge algorithm reaches 99%.

The rest of this chapter is organized as follows. Section 5.2 formulates the problem and emphasizes the focus of our study on merging trees that correspond to different ingresses. Section 5.3 first briefly discuss the tree construction algorithm borrowed from previous studies and then detail the design of the tree merge techniques. Section 5.4 evaluates the performance of tree construction and the merge algorithm via extensive simulation on a variety of network conditions, and Section 5.5 presents the evaluation of the proposed techniques under real Internet experiments. Section 5.6 discusses the related work, and this chapter concludes with Section 5.7.

5.2 Problem Statement

The knowledge of the network topology is critical for network anomaly detection and troubleshooting. Ideally, with the complete knowledge of the topology, the origin of each anomaly can be easily pinpointed by correlating the conditions on paths between each pair of ingress and egress across the network. Traditional measurement tools, such as traceroute, which require cooperation from the network devices, can be used to obtain the list of routers between the source and the destination in consideration. The effectiveness of traceroute, however, greatly hinges on the responsiveness of the network devices. Unfortunately, such condition becomes less common as the concerns with the increase of malicious activities, *e.g.,* denial-of-service attacks, Internet worms. In particular, the unification of Layer-2 (*e.g.,* MPLS) and Layer-3 IP technologies could render the path information hidden in lower layer. It is thus important to be able to infer topological information without cooperation from the individual devices. In our study, we only assume that availability of end-to-end performance measurements across each network, as such measurement data can obtained from end-host based probing alone [104]. Nowadays, each ISP specifies in its SLA (Service Level Agreement) how traffic is treated while traversing its network. It should also provide ways to allow its customer network to obtain the performance of the cross traffic to ensure the quality of service advertised in the SLA is achieved. Our work provides a way to obtain such information without depending ISP's help.

The identification of network topology based on end-to-end performance measurements has been studied extensively in the area of network tomography and a number of techniques [97, 98, 99, 100, 101, 102, 103] have been proposed to infer the tree structure connecting a single sender (the ingress in our discussion) to multiple receivers (the egresses). These studies solely rely on end-to-end measurements and do not require the knowledge of the individual devices in the network, and therefore, it is impossible to identify the complete physical topology. Rather, a *logical* topology defined by the branching points between paths to different receivers is obtained. As shown in Figure 5.2(a), the physical topology is represented as a directed graph, where each vertex represents a physical device (*e.g.,* a router or a switch) and the edges correspond to the connections between these devices. In contrast, Figure 5.2(b) is the corresponding logical topology. In the logical tree, each vertex represents a physical network device where traffic branching occurs, *i.e.,* where two

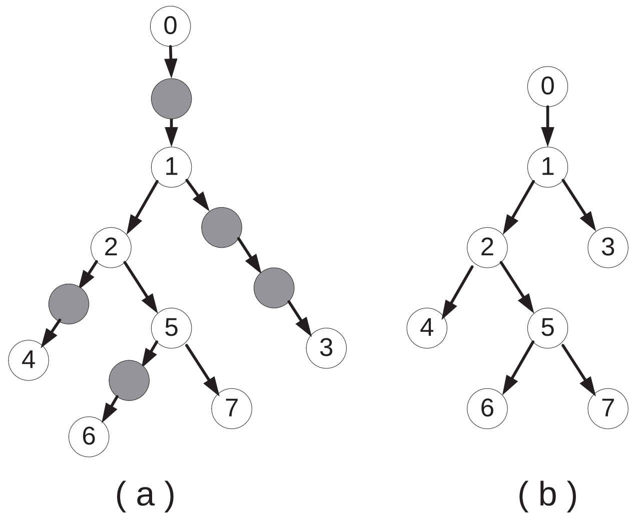

Figure 5.2: A Physical topology in (a) and the corresponding logical topology in (b).

or more paths from the sender to the receivers diverge. The grey nodes in Figure 5.2(a) cannot be identified because the traffic through each of these devices does not diverge to reach receivers underneath and end-to-end measurements cannot be used to distinguish it from other devices along its branch.

Unfortunately, previous studies have mostly concentrated on discovering the tree topology connecting a single source to multiple receivers and the problem of correlating trees rooted at different sources have not been addressed. Our study aims to fill this void by proposing techniques that merge together trees corresponding to different sources to obtain a more complete network topology. In what follows, we describe in details how to identify resource sharing across ingress and egress of a single ISP by only using the end-to-end performance measurements across the network.

5.3 The Proposed Approach

As described earlier, the goal of our study is to identify resource sharing across different ingress and egress points in an ISP. To achieve this goal, we divide the task into the following two intuitive steps. First, we construct tree structures for each ingress from the end-to-end measurement data collected from the ingress to all of the egress nodes. Second, we merge trees rooted at different ingress nodes to compute a more complete topology for the network which can be used to identify the resource sharing across different ingress nodes.

Our proposed techniques are based on the following reasonable assumptions.

A1. The underlying topology is *fixed* during each measurement. The internal topology of large ISPs are shown to be stable within tens of minutes to hours. The end-to-end measurements are thus collected accordingly to ensure the validity of the assumption.

A2. The underlying topology connecting a single sender to multiple receivers is a *tree*. Networks that use shortest-path-based routing satisfy the assumption. Meanwhile, we notice that network devices apply load balancing [105] to forward packets via separate routes based on the source address, the destination address, and even other fields in the packet header. We seek to fix these fields as much as possible to minimize their effects on the topology.

5.3.1 The Tree Construction

A variety of techniques have been proposed in network tomography to infer the tree-structured network topology. Here we discuss a method called "hierarchical clustering" [99], which performs well with little computation overhead. The general hierarchical clustering algorithm works as follows.

1. Choose the pair of nodes with the highest similarity;
2. Merge the pair into a new node;
3. Update the similarities between the new node and the previously existing nodes; and
4. Repeat the procedure until only one node is left.

Apparently, by continuously applying the clustering algorithm to the set of receivers as the initial set of nodes, we eventually reach a tree structure where the last remaining node is the root node. To illustrate the principle of this process, let's consider the logical tree in Figure 5.2(b). Suppose with each internal node K in the tree we associate a metric M_K which is measured between root 0 and node K. For example, the packet delay from 0 to K. Here we only consider metrics that have the following *monotonic* property: An internal node has a smaller metric than any of its descendants, *e.g.*, $M_5 > M_2 > M_1$ in the figure. Metrics that satisfy such a property include loss rate, delay, and delay-variance. Since the topology is unknown, the metric for the

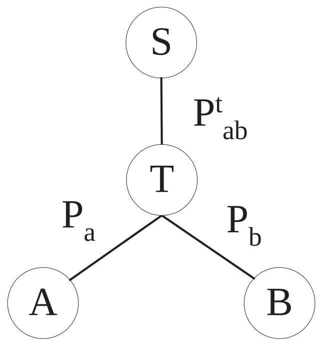

Figure 5.3: Binary tree loss model.

internal nodes has to be estimated from the leaf nodes, *i.e.*, the end-to-end measurements. Let $a(i, j)$ denote the nearest common ancestor of a given pair of nodes i, j, *e.g.*, $a(4, 7) = 2$. Define $M(i, j) = M_{a(i,j)}$. The value of $M(i, j)$ can be regarded as a characterization of the shared portion of the paths from the root 0 to i and j, *i.e.*, the path from the root 0 to node $a(i, j)$.

In the context of the hierarchical clustering algorithm, $M(i, j)$ can be interpreted as the similarity measures of i and j. Note that the metric exhibits a symmetric feature, $M(i, j) = M(j, i)$; that is, the ordering of receivers does not affect the computed metric. The monotonicity property of the metric ensures the identification of the tree topology via the hierarchical clustering algorithm. For example, in the figure, $M(6, 7)$ is greater than any other $M(i, j)$, revealing that nodes 6 and 7 are the deepest level of the tree and must first be merged together in the clustering process. The tree topology can thus be recursively constructed by applying the clustering algorithm each step of which finds two nodes with the highest similarity (*i.e.*, $M(i, j)$) to merge.

Since tree construction is not the focus of our study, we borrowed the hierarchical clustering idea in [103] to infer each tree structure based on end-to-end measurements of the loss rate. The technique assumes that the measurements are collected in multicast communications. Figure 5.3 illustrates a binary tree loss model with $P^t_{a,b}$ as the probability of packet loss between the source S and the internal node T, and P_a, P_b as the probability of packet loss between T and A, B, respectively. Assume that the packet loss probabilities on different links in the tree are independent. In the scheme, each receiver X (*e.g.*, A, B) records a loss sequence L_x which is an ordered list of the sequence numbers of packets lost by X. Let's consider the loss sequences of A and B. Any packet lost along their shared path between S and T will appear in both L_A and L_B, which we call "true" shared losses. In addition to these true shared losses, it is possible that two copies of the same packet are lost independently along their distinct paths, between T and A, and between T and B. These are called "false" shared losses. Let P_{ab} be the probability of seeing a shared loss (whether true or false) between A and B. Then

$$P_{ab} = P^t_{ab} + (1 - P^t_{ab}) \times P_a \times P_b.$$

Let the probability of seeing a loss at A but not at B be

$$P_{a\bar{b}} = (1 - P^t_{ab}) \times P_a \times (1 - P_b).$$

Similarly,

$$P_{\bar{a}b} = (1 - P^t_{ab}) \times (1 - P_a) \times P_b.$$

Solving the three equations above yields P^t_{ab}, P_a, and P_b:

$$P^t_{ab} = \frac{P_{ab} \times P_{\bar{a}b} + P_{\bar{a}b} \times P_{a\bar{b}} + P_{a\bar{b}} \times P_{ab} + P^2_{ab} - P_{ab}}{P_{ab} + P_{\bar{a}b} + P_{a\bar{b}} - 1}$$

$$P_a = \frac{P_{a\bar{b}}}{1 - (P_{\bar{a}b} + P_{ab})}$$

$$P_b = \frac{P_{\bar{a}b}}{1 - (P_{a\bar{b}} + P_{ab})}.$$

Note that P_{ab} can be measured by $|L_{ab}|/N$, where $L_{ab} = L_a \cap L_b$ and N is the total number of probing packets. Similar measures can be applied to get $P_{a\bar{b}}$ and $P_{\bar{a}b}$. The binary tree can be constructed as follows: (1) compute the probability of "true" shared losses between all pairs of receivers; (2) choose the pair of receivers with the maximum probability and combine them together into an internal node, updating the loss sequence as $L_{ab} = L_a \cap L_b$; (3) repeat the two steps until only one node is left, which is the root of the tree. Algorithm 1 presents the pseudo-code for the binary tree construction algorithm.

The tree constructed so far is a binary tree and the algorithm can be extended to build an arbitrary tree with small modifications. Figure 5.4 presents an example in which a binary tree can

```
*function* **construct_tree**($R$, $L$)
# $R$: set of receivers. $R = \{R_i | i = 1, 2, \ldots, N\}$
# $L$: set of loss sequences $L_i$ for $R_i$. $L = \{L_i\}$
1:  # initial set of unconstructed nodes
2:  $S = \{S_i = R_i | i = 1, 2, \ldots, N\}$;
3:  **while not** empty($S$) **do**
4:      **for all** $S_i$, $S_j$ ($i \neq j$) $\in S$ **do**
5:          Compute $P^t_{i,j}$ from $L_i, L_j$;
6:      **end for**
7:      # find the two nodes to combine
8:      $P^t_{a,b}$ =*MAXIMUM*($P^t_{i,j}$), for any $S_i, S_j (i \neq j) \in S$;
9:      # combine $S_a$, $S_b$ and replace them by $S_c$;
10:     $S = (S \setminus \{S_a, S_b\}) \cup \{S_c\}$;
11:     $L_c = L_a \cap L_b$; # update loss sequence
12: **end while**
```

Algorithm 1: Tree construction

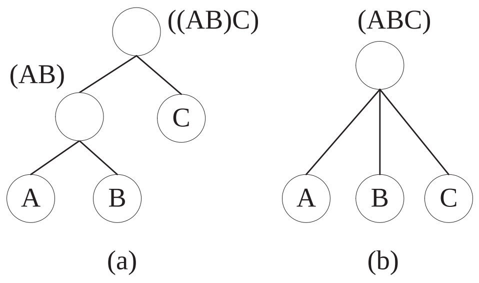

Figure 5.4: A binary tree in (a) vs. a ternary tree in (b)

be coalesced into a ternary tree. Here we are building a tree connecting three receivers A, B, and C. When the binary tree can be converted into the ternary tree, the following condition must be satisfied: $P^t_{(ab)c} = P^t_{ab}$. Therefore, when we merge node (AB) with C, we compare the value $P^t_{(ab)c}$ with P^t_{ab}. If they are identical or within a small difference range specified by a threshold, we can coalesce the binary tree into the ternary tree. From the above discussion, we can see the binary tree is the most fundamental tree structure, and other type of the tree can be easily converted to a binary tree.

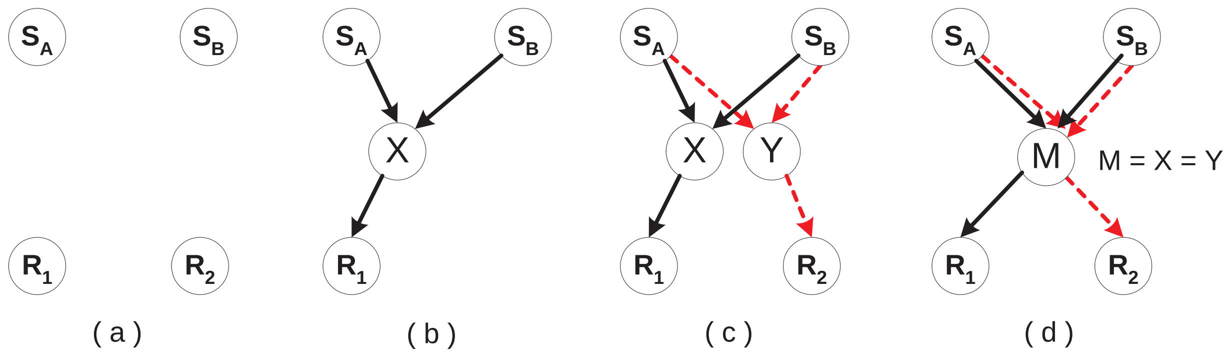

Figure 5.5: The merge procedure in a two-sender-two-receiver network. (a) is the network before merge; (b) presents the merge for one receiver; (c) is the merge for both receivers; (d) is the scenario when the merge of both receivers converges.

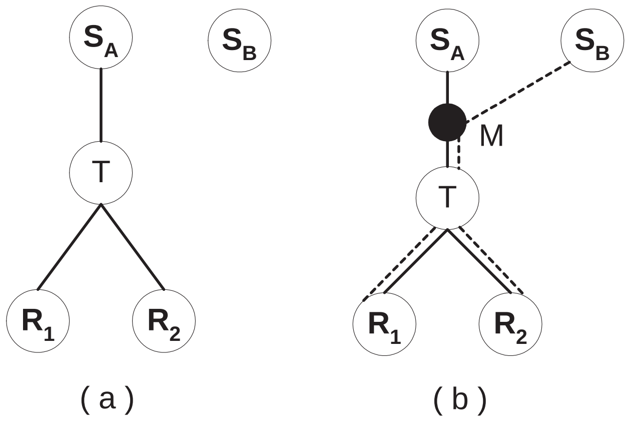

Figure 5.6: A tree merge example. (a) topology before merge; (b) topology after merge.

5.3.2 The Tree Merge

After constructing trees for each ingress node, we next merge trees together to produce the more complete topology of the network and identify the resource sharing across different ingress and egress nodes.

Before we discuss the merge across trees, we first look at a simple two-sender-two-receiver network and investigate the fundamental features that allow us to determine the underlying topology. Figure 5.5 presents the entire merge procedure. Without prior topological knowledge of Figure 5.5(a), we consider two receivers separately. Figure 5.5(b) shows the scenario in which the path from S_B to R_1, P_{S_B,R_1}, converges somewhere along the path from S_A to R_1, P_{S_A,R_1}. Unfortunately, the merge point X, which is located between S_A and R_1, cannot be precisely pinpointed by relying solely on the end-to-end performance measures. Similarly, it applies to the second receiver R_2 as shown in Figure 5.5(c). However, in case the merge locations for R_1 and R_2 coincidentally converge, *i.e.,* $X = Y$, we can further scale down the possible merge locations. As shown in Figure 5.5(d), the merge converges at a common location M. M, previously (*i.e.,* X or Y) arbitrarily located on P_{S_A,R_1} and P_{S_A,R_2}, now has to be located on the *shared* path between P_{S_A,R_1} and P_{S_A,R_2}. Figure 5.6 presents the merge of S_B's tree into S_A's tree. Here we assume the merge of two receivers converges at M. As discussed earlier, the dark circle M in Figure 5.6(b) must be located at the commonly shared segment of paths to R_1 and R_2, *i.e.,* the path between S_A and T.

The key question now is to decide whether the merge across receivers should converge based on the end-to-end performance measures, denoted by M. Suppose the metric M has the *additive* property, *i.e.,* the measure of a path is the sum of individual measures of the links along the path. Accordingly, the measure from S_A to R_1,

$$M_{S_A,R_1} = M_{S_A,X} + M_{X,R_1},$$

and the measure from S_B to R_1,

$$M_{S_B,R_1} = M_{S_B,X} + M_{X,R_1}.$$

Obviously, the measures from two senders have a common contributing factor M_{X,R_1}. The discrepancy of the measures from two senders, $D_{R_1}^{S_A,S_B}$, can thus be calculated as follows:

$$D_{R_1}^{S_A,S_B} = M_{S_A,R_1} - M_{S_B,R_1} = M_{S_A,X} - M_{S_B,X}.$$

Similarly, for the other receiver R_2, we have

$$D_{R_2}^{S_A,S_B} = M_{S_A,R_2} - M_{S_B,R_2} = M_{S_A,Y} - M_{S_B,Y}.$$

When X and Y coincide at M, we know that the differences of performance measures from senders S_A and S_B across receivers R_1 and R_2 are equal, *i.e.,*

$$D_{R_1}^{S_A,S_B} = D_{R_2}^{S_A,S_B} = M_{S_A,M} - M_{S_B,M}.$$

We also noticed $X \neq Y$ from in Figure 5.5(c), so we cannot rule out the possibility that a certain network setup would make the end-to-end performance discrepancies identical. Therefore, the discrepancies of end-to-end measures being equal is a *necessary* but *not sufficient* condition for the convergence of the merge locations across different receivers.

Despite the fact the identical end-to-end performance discrepancies do not always guarantee the coincidence of the merge locations for the corresponding receivers, it is still a good indicator. Given usually fluctuating network conditions, it is extremely unlikely that topologically-uncorrelated paths (*e.g.*, paths from the senders to X and Y in Figure 5.5(c)) exhibit identical behaviors over a long-term period. In Section 5.3.2.B, we also propose to use multiple metrics together to enhance the "sufficiency" of identical performance discrepancies for the coincidence of merge locations.

A. The Algorithm

We now formally present the merge algorithm which generalizes the merge in a two-receiver-two-sender network to that in an N-receiver two-sender network, where $N > 2$. In Section 5.3.2.D, we briefly discuss the algorithm for multiple-sender scenarios.

Suppose the sender S_B's tree is to be merged into the sender S_A's tree. The merge algorithm can be divided into two steps. In the first step, the algorithm first calculates for each receiver the performance discrepancies from the two senders, and then clusters together the receivers based on the *similarity* of their performance discrepancies. Algorithm 2 presents the pseudo-code of the process. Lines 1–5 compute for each receiver the performance difference across the two senders. Lines 6–8 calculates the absolute difference between any of two receivers in their performance discrepancies. We are interested in those receivers that observe identical performance discrepancies across two senders. Therefore, we focus on $D_{i,j}$ with a value close to zero. Lines 9–25 essentially describe the procedure in which the set of receivers are partitioned into separate clusters each of which contains receivers with very similar end-to-end performance discrepancies. In Line 13, a threshold value, which is close to zero, is set to distinguish receivers across clusters. In general, setting the threshold too small would separate intra-cluster receivers into different clusters while setting the threshold too large would group uncorrelated receivers into a cluster. In Section 5.4.4, we elaborate on the selection of the threshold via our simulation-based analysis.

```
function cluster_receivers($R$, $S_A$, $S_B$)
# partition the set of receivers into sub-clusters;
# $R$: set of receivers. $R = \{R_i | i = 1, 2, \ldots, N\}$
# $S_A$, $S_B$: two senders.
 1: for all $R_i \in R$ do
 2:     $C_i = \{R_i\}$; # initial clusters
 3:     # $M_i^A$, $M_i^B$: performance measure from $S_A$,$S_B$ to $R_i$
 4:     $D_i = M_i^A - M_i^B$; # compute performance difference
 5: end for
 6: for all $R_i, R_j \in R$, $i \neq j$ do
 7:     $D_{i,j} = |D_i - D_j|$; # compare difference across receivers
 8: end for
 9: $L = \{D_{i,j} | i \neq j, i, j = 1, 2, \ldots, N\}$;
10: Sort $L$ in an increasing order;
11: while not empty($L$) do
12:     Remove the first item $D_{i,j}$ from $L$;
13:     if $D_{i,j} >$ THRESHOLD then
14:         break;
15:     end if
16:     $C = C_i \cup C_j$; # merge two clusters
17:     for all $R_k \in C_i$, $R_l \in C_j$ do
18:         Remove $D_{k,l}$ from $L$; # update $L$
19:     end for
20:     for all $R_k \in C$ do
21:         $C_k = C$; # update clusters
22:     end for
23: end while
24: $C = \{C_i | i = 1, 2, \ldots, N\}$;
25: $C$ = uniq($C$); # eliminate identical items
26: return $C$;
```

Algorithm 2: Cluster receivers based on their performance discrepancies

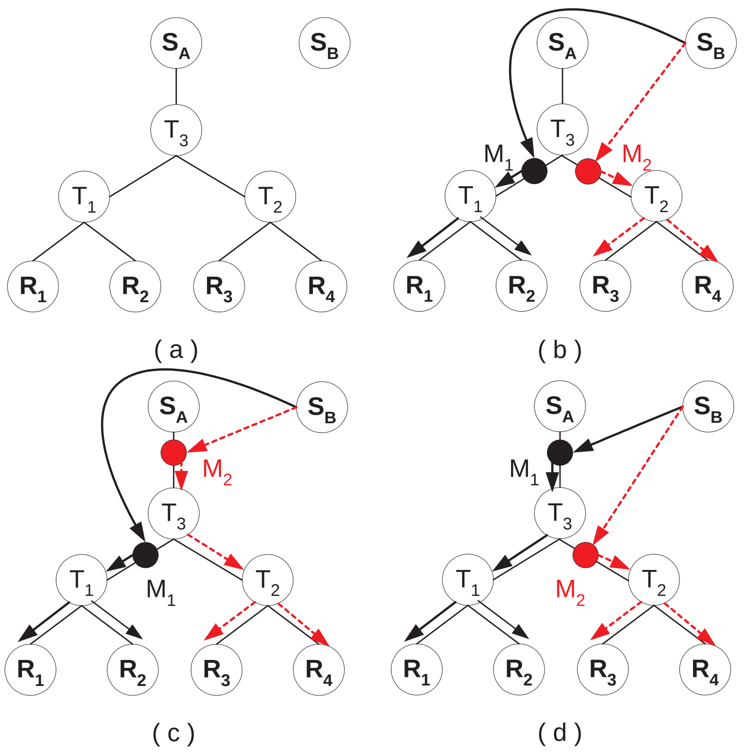

Figure 5.7: A tree merge example. (a) topology before merge; (b–d) three possible topologies after merge.

The first step of the merge algorithm produces several clusters of receivers. So far, the algorithm has not dealt with any of the tree structures rooted at S_A or S_B. In the second step, we pinpoint for each receiver its possible merge locations in S_A's tree by applying the two following heuristics.

Heuristic I: For receivers within a common cluster, their merge locations from S_B into S_A's tree must converge. As illustrated in Figure 5.6(b), the first heuristic determines the set of potential merge locations in the tree rooted at S_A by *intersecting* the paths from S_A to the receivers in the same cluster.

Let us look at a more complicated example in Figure 5.7. Suppose the set of receivers are partitioned into two clusters, $\{R_1, R_2\}$ and $\{R_3, R_4\}$. According to Heuristic I, for R_1 and R_2, the merge location from S_B into S_A's tree is in the path $[S_A, T_3, T_1]$ while the merge locations of R_3 and R_4 are in the path $[S_A, T_3, T_2]$. Unfortunately, we cannot further reduce the set of merge

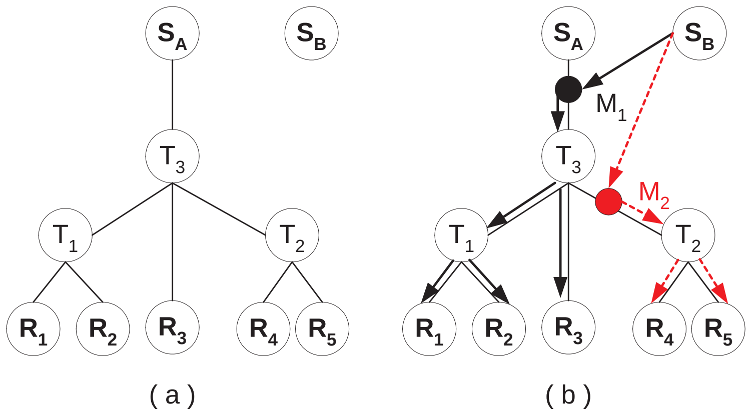

Figure 5.8: A tree merge example demonstrating merge rule 2. (a) topology before merge; (b) topology after merge.

locations for these receivers as Figure 5.7(b–d) presents three possible merge combinations for four receivers. In the figure, M_1 is the merge location for R_1 and R_2, and M_2 is the merge location for R_3 and R_4. The only merge combination we can rule out is that M_1 and M_2 cannot both be between S_A and T_3; otherwise, the four receivers would have equivalent performance discrepancies across S_A and S_B, and hence would be grouped in a single cluster. That is, M_1 being in path $[S_A, T_3]$ precludes M_2 from being in $[S_A, T_3]$, and vice versa. This observation inspires our second heuristic.

Heuristic II: The set of merge locations for a cluster of receivers *must not* be a subset of the set of merge locations for another cluster of receivers.

We illustrate this heuristic by the example in Figure 5.8. Suppose we have two separate clusters after the first step: $\{R_1, R_2, R_3\}$ and $\{R_3, R_4\}$. Applying Heuristic I, M_1, the merge point for R_1, R_2, and R_3, is in path $[S_A, T_3]$, and M_2, the merge point for R_4 and R_5, is in path $[S_A, T_3, T_2]$. M_1 and M_2 cannot be at the same location; otherwise, the five receivers would have been in one cluster. Therefore, the merge point M_2 cannot be in path $[S_A, T_3]$, and it has to be in path $[T_3, T_2]$. Heuristic II can then be translated as follows: if one set of merge locations is a subset of merge locations for a different cluster of receivers, subtract the subset from the superset.

The pseudo-code of the complete tree merge algorithm is presented in Algorithm 3. Lines 3–10 and 11–16 are implementations of two merge heuristics, respectively.

```
function merge_tree($R$, $S_A$, $S_B$)
# merge $S_B$'s tree into $S_A$'s with respect to $R$
# $R$: set of receivers. $R = \{R_i | i = 1, 2, \ldots, N\}$
1:  # cluster receivers based on their end-to-end performance
2:  $C$ = cluster_receivers($R$,$S_A$,$S_B$);
3:  # Merge Heuristic I
4:  for all $C_k \in C$ do
5:      for all $R_i \in C_k$ do
6:          Compute the path $P_i^A$ from $S_A$ to $R_i$;
7:          $P_k = P_k \cap P_i^A$; # compute the common path segment
8:      end for
9:      $P_k$ is the set of possible merge locations for $R_i \in C_k$;
10: end for
11: # Merge Heuristic II
12: for all $P_k$, $P_l$, $l \neq k$ do
13:     if $P_l \subset P_k$ then
14:         $P_k = P_k \setminus P_l$;
15:     end if
16: end for
```

Algorithm 3: Merge trees across two senders

B. Selection of Metrics

The merge algorithm requires that the end-to-end performance metric be "additive". Meanwhile, the tree construction is also based on performance metrics with the monotonicity property. The loss rate, delay, and delay-variance all satisfy the property. The delay and delay-variance are additive. The loss rate, however, has to be converted to a different form before satisfying the additive property. Let us consider Figure 5.3, the probability of packets being successfully delivered from S to A is the multiplication of the probability of successful packet delivery on link $[S, T]$ and that on link $[T, A]$. By taking the logarithmic form, the metric measuring the end-to-end successful packet delivery becomes an additive performance metric.

C. Validating the Constructed Tree

As described earlier, the first step of the merge algorithm does not rely on the structure of the constructed tree. Conversely, the tree merge process sometimes could help verify the correctness of the constructed tree. Here we assume that the set of receivers are correctly partitioned into clusters. According to the rationale behind the clustering process, receivers in different clusters cannot merge at the same location. This property can then be used to validate the tree structure. Let us consider the topology in Figure 5.8. Suppose the clustering process decides on the following two clusters $\{R_1, R_4, R_5\}$ and $\{R_2, R_3\}$. It contradicts with S_A's tree as the set of merge locations for R_1, which is in path $[S_A, T_3]$, cannot be isolated from the set of merge locations for R_2 and R_3, which is also in path $[S_A, T_3]$.

D. Selection of Senders

In Section 5.3.2.A, we do not distinguish between S_A and S_B. In general, merging S_A's tree into S_B's tree should not differ from merging S_B's tree into S_A's tree. However, considering the two heuristics in pinpointing the potential merge locations, we prefer to select a reference tree (*i.e.*, S_A's tree in Algorithm 3, which is used for the computation of merge locations) which has more sharing, or higher branching factors for non-leaf nodes in the tree.

The merge algorithm described so far deals with the merge between two senders' trees. It can also be generalized to be applicable for merging trees across more than two senders. First, we select a tree as a reference tree. Then, we sequentially apply the merge algorithm in Algorithm 3 and merge each of the remaining senders' trees into the reference tree.

5.3.3 Identifying Resource Sharing

Relying on the limited data from the end-to-end performance measures, the tree construction and merge algorithm is impossible to identify the complete topology for the network. However, we still acquire some knowledge about the sharing of the resources in the network which can thus be used in network diagnosis and to suggest measures to alleviate impacts caused by failures, congestion, etc.

As shown in Figure 5.9, S_B and S_C's trees are merged into the tree rooted at S_A. S_B's tree

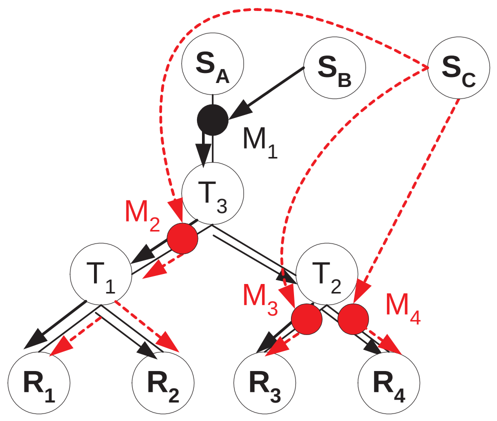

Figure 5.9: Identification of resource sharing

merges at M_1 while S_C's tree joins S_A's tree at M_2, M_3, and M_4. For each sender, by correlating the conditions across the set of receivers with the constructed tree, we are able to scale down the origin of the anomaly. Moreover, by having the knowledge of the merge locations across trees associated with different senders, we can come up with a more feasible solution to divert the impact of the problem. For example, suppose failure or congestion renders the poor performance on the link between S_A and T_3. According to the inferred merge scenario, traffic which used to enter the network via S_A can then be diverted to S_C rather than S_B to bypass the problem.

5.4 Simulation

In this section, we evaluate our proposed inference techniques via simulation. We use ns2 simulator [106] to test the performance of the algorithms under a variety of network scenarios. In the following, we first describe the simulation setup, then we discuss the simulation results on the tree construction algorithm. Lastly, we evaluate the accuracy of the tree merge algorithm.

5.4.1 Simulation Setup

In order to fully test the performance of the tree construction and merge algorithm, we vary several parameters to generate various simulation scenarios. In each of ns2 simulation runs, the topology is based on binary tree structures, that is, the tree rooted at each sender is a binary tree. As

Loss condition	**On/off ratio**	**End-to-end loss rate**
good	0.1	0.5% - 1.0%
fair	0.2	1.0% - 2.0%
bad	1.0	5.0% - 10.0%

Table 5.1: Loss configuration in simulations

discussed in Section 5.3.1, any non-binary tree can be easily converted into a binary tree by splitting each node with more than two branches with extra superfluous nodes with metrics equivalent with the original node. Using binary trees does not invalidate the evaluation of the proposed techniques. Each link in the topology is equipped with a randomized capacity of 1Mb/sec to 5Mb/sec and latency of 10ms to 30ms to simulate the heterogeneity present in a large ISP. Probing traffic was generated from each sender independently with a constant rate of 20 packets per second. We also generate background traffic which is comprised of a mix of FTP (over TCP) and CBR (over UDP) traffic. The source and the destination of these background traffic flows are randomly picked and their durations are controlled by an exponential on-off model. The following parameters are varied in each test to investigate the performance of the algorithms

- **# of Receivers:** The number of receivers essentially determines the size of the entire binary tree topology in each simulation run. Specifically, the number of receivers is set to 7, 15, or 30 and the total number of nodes in a corresponding two-sender topology is 17, 35, and 66, respectively.

- **Loss Rate:** In our simulation, the generation of the loss rate is two-folded. On one hand, the packet loss can be due to buffer overflow at the queue associated with each link. On the other hand, we associate a randomly selected link with a uniform packet-dropping loss model. The selection of the random link is determined by a single on-off exponential model and the uniform loss model also randomly selects a loss rate ranging from 50% to 80%. By choosing the high loss rate, we focus on simulating network scenarios such as link failure and network congestion. For each individual receiver, the end-to-end loss rate is controlled by adjusting the parameters in the on-off model. Table 5.1 presents the three loss conditions: “good”, “fair”, and “bad” with corresponding ratio between the on-time and the off-time and

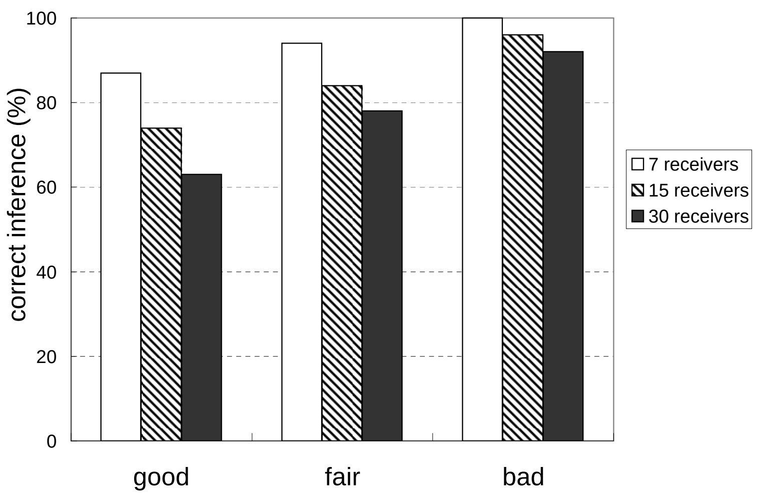

Figure 5.10: Tree inference accuracy

the end-to-end loss rate.

- **Communication Scheme:** The tree construction algorithm in Section 5.3.1 assumes the probing traffic utilizes the underlying multicast scheme. Realizing that the IP multicast has not been widely deployed, our simulation also tests how the tree inference works when the probing packets are sent to individual receivers via unicast.

5.4.2 Tree Construction

The first set of simulation runs evaluate the performance of the tree construction algorithm. The probing traffic is multicast to the set of receivers. We test 100 runs for each of the nine possible combinations of the topology size and the loss condition. The performance of the algorithm is measured by the percentage of correctly inferred trees and Figure 5.10 presents the results. As we can see, the inference accuracy increases with the worsening of the loss condition. For the case with 7 receivers, it achieves 87% inference accuracy under the "good" condition while reaching 100% under the "bad" condition. This can be explained by the rationale behind the tree inference process. When the loss condition worsens, the loss probability on each link increases and the shared loss probability for deep nodes in the tree thus can be more discernible from that for the shallow ones. The clustering procedure matches better with the actual tree structure.

Meanwhile, with the increase of the topology size, the inference accuracy decreases. Under

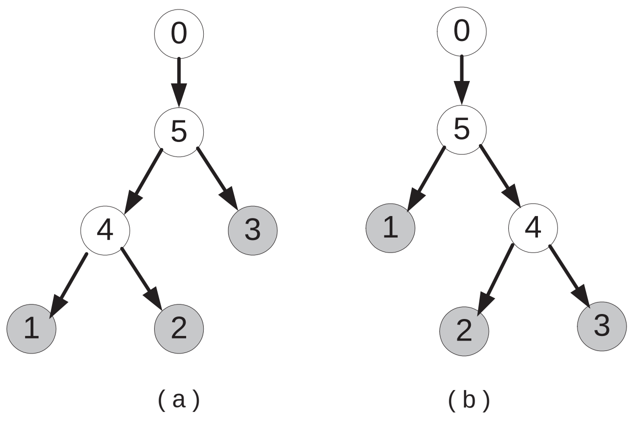

Figure 5.11: Quantifying the mismatch of incorrectly inferred tree: (a) the actual tree; (b) the inferred tree

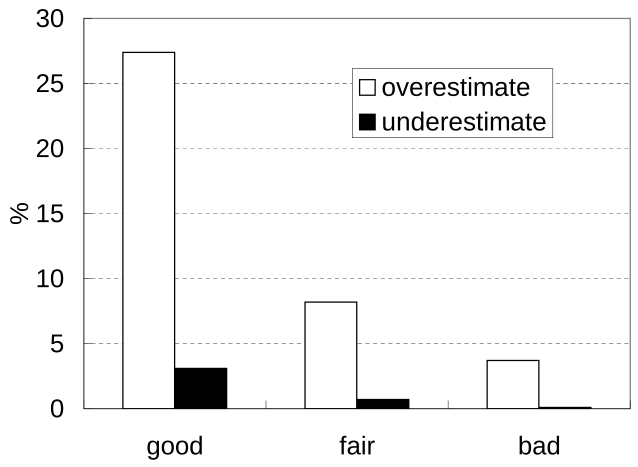

Figure 5.12: Mismatches of the incorrectly inferred tree (30 receivers)

the "good" condition, the algorithm can only correctly identify 63% of the topologies with 30 receivers. This can be attributed to two factors. First, with more nodes to cluster, the possibility of the misstep during the recursive clustering process, especially when the loss probability on each individual link becomes small. This is confirmed by the fact that the inference accuracy gap for different topology sizes becomes wider when the loss condition improves. Second, in our simulation, the loss model, controlled by a single exponential on-off process, is randomly applied to the links. A large network has more links and each link thus becomes less likely to be selected. Therefore, the overall loss rate becomes smaller and the inference accuracy further decreases.

To further evaluate the cases when the inferred topology mismatches with the actual tree, we

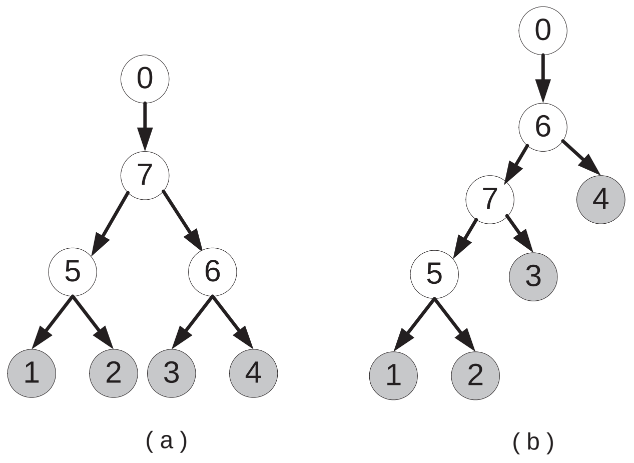

Figure 5.13: An incorrectly inferred tree. (a) the actual tree; (b) the inferred tree

define a new metric to capture the notion of mismatch between the two trees. In particular, for a specific tree, we consider how each receiver fares with respect to the rest of the receivers. We define $\lambda_{i,j}$ as the number of shared links between receiver R_i and receiver R_j. Figure 5.11 illustrates an example when the inferred tree does not match the actual tree. By using $\lambda_{i,j}$, the actual tree has $\lambda_{1,2}$, $\lambda_{1,3}$, and $\lambda_{2,3}$ being equal to 2, 1, and 1, respectively, while the corresponding measures in the inferred tree are 1, 1, and 2, respectively. In Section 5.3.3, the decision-making in the proactive network troubleshooting largely depend on the degree of the resource sharing among paths. $\lambda_{i,j}$ to some degree reflects the resource shared among R_i's path and R_j's path. Comparing the actual tree in Figure 5.11(a), tree in Figure 5.11(b) *underestimates* the sharing between node 1 and node 2 (a smaller $\lambda_{1,2}$) while *overestimating* the sharing between node 2 and node 3 (a larger $\lambda_{2,3}$).

Figure 5.12 shows the mismatches of the incorrectly inferred trees from the actual tree for the case of 30 receivers. We count the number of underestimates and overestimates and compute their percentage relative to the total number of $\lambda_{i,j}$ for the set of receivers (it is $\frac{30\times29}{2} = 435$). First, as the loss condition becomes worse, the mismatches between the inferred tree and the actual tree diminishes, which exhibits the same trend as in Figure 5.10 because higher loss rate improves the accuracy of the tree inference. Second, we notice that the percentage of overestimates is more significant than the percentage of the underestimates of the sharing. This can be explained by the example shown in Figure 5.13. By computing $\lambda_{i,j}$, the inferred tree has 3 overestimates but only 1 underestimates. A careful comparison of the trees reveals that the inferred tree only underestimates

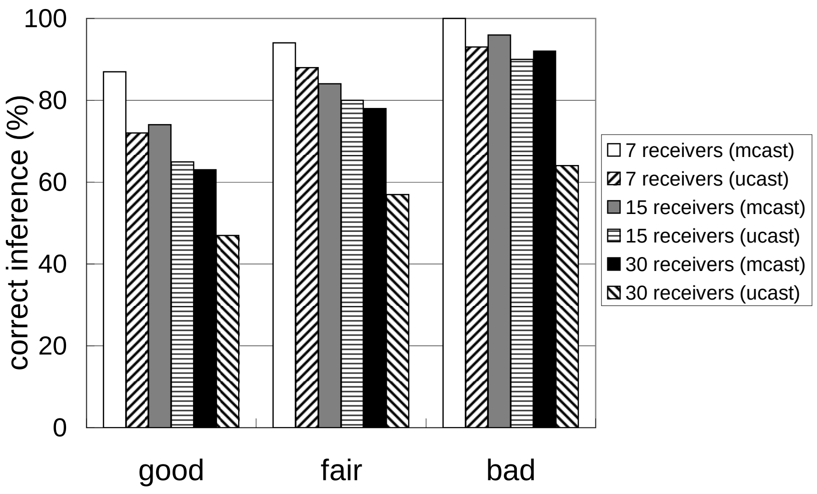

Figure 5.14: Tree inference accuracy (multicast vs. unicast)

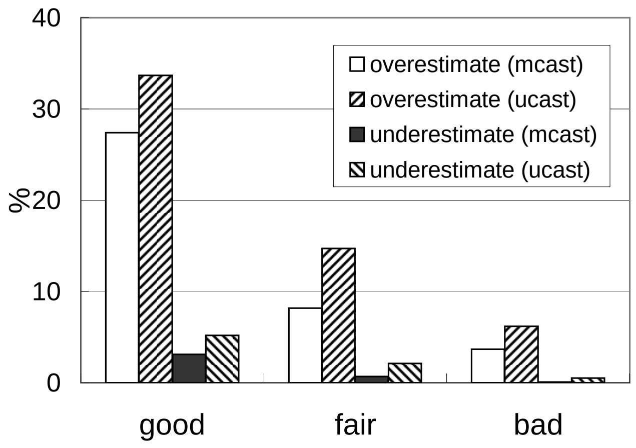

Figure 5.15: Mismatches of the incorrectly inferred tree for 30 receivers (multicast vs. unicast)

the sharing between node 3 and 4. The number of overestimates is artificially exaggerated because the inferred tree is one layer deeper than the actual tree. Therefore, it is more important to ensure that the percentage of the underestimates is small, which is also in line with our ultimate goal because the selected path in Figure 5.9 must never underestimate its sharing with other paths.

5.4.3 Multicast vs. Unicast

The tree inference algorithm assumes the probing packets are sent to the receivers via multicast. Notice that the IP multicast has not been widely deployed in today's Internet, we must consider the scenario where only unicast is permitted. In this section, we contrast the performance of the tree inference algorithm under unicast and multicast. Figure 5.14 presents the inference accuracy under both schemes and Figure 5.15 illustrates the mismatches of the incorrectly inferred tree. As we can see, the inference accuracy suffers when applying the algorithm to the unicast probing packets. The situation becomes worse when the loss rate is small. In addition, for large topology, the increase of loss rate does not seem to improve the percentage of accurately inferred topology in the same pace of the smaller topologies. The explanation is that, with a large graph, each individual unicast packet tends to be affected more as it traverse the network to reach the receiver and the correlation among unicast packets decreases greatly compared with the multicast packets.

5.4.4 Tree Merge

As described in Section 5.3.2, the critical step of the tree merge algorithm is to correctly partition the set of receivers into clusters based on their end-to-end performance discrepancies across the senders. In this section, we first elaborate some practical concerns in manipulating the metrics (*e.g.*, loss rate, delay, *etc.*) for the merge purposes. Then, we evaluate the accuracy of the tree merge techniques.

When using the measures of loss rate as the metric, in addition to choosing an appropriate mathematical form as described in Section 5.3.2.B, we also must select a proper time window to calculate the loss rate. Choosing a small time window magnifies its fluctuation inherent in the real networks and thus enlarges the difference $D_{i,j}$ in Algorithm 2 even for receivers that are supposed to be in the same cluster, resulting in false negative decision. By choosing a large time window, on the other hand, the unique loss pattern for each receiver might be lost and $D_{i,j}$ for receivers not in the same cluster becomes small, *i.e.,* false positives. In our simulation, the time window is selected in such way that it is a small multiple of the on-time period which defines the rate in which the lossy condition moves around the links.

Second, $D_{i,j}$ in Algorithm 2, taking the absolute form, does not work well because it is affected by the value of D_i and D_j. In practice, it is better to compute a relative metric $D_{i,j}^{relative}$.

$$D_{i,j}^{relative} = MAX(\frac{D_{i,j}}{|D_i|}, \frac{D_{i,j}}{|D_j|})$$

Third, the selection of the threshold in Algorithm 2 Line (13) is critical. In our analysis, we

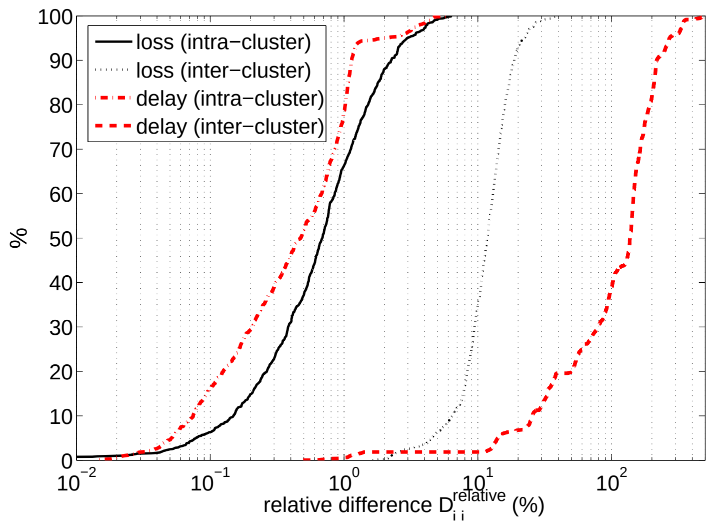

Figure 5.16: CDF of loss and delay difference in tree merging

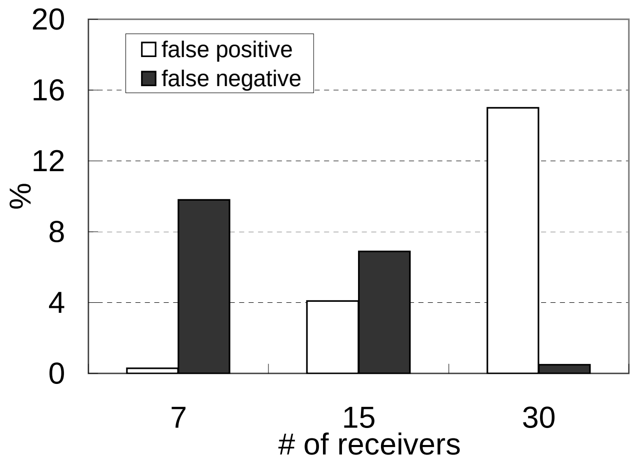

Figure 5.17: Clustering accuracy in tree merging (delay)

compute $D_{i,j}^{relative}$ for both the pair of intra-cluster receivers and the pair of inter-cluster receivers and Figure 5.16 presents the cumulative distribution function of $D_{i,j}^{relative}$ for intra-cluster and inter-cluster receivers by using loss rate or delay. As shown in the figure, for both the delay and loss rate, these is no perfect value for the threshold such that it separates the intra-cluster receivers from the inter-cluster receivers. Instead, in our analysis, we choose 2% and 5% as the threshold for the metric in the loss rate and the delay, respectively.

We first evaluate the accuracy of clustering process when delay is the metric. Because the loss

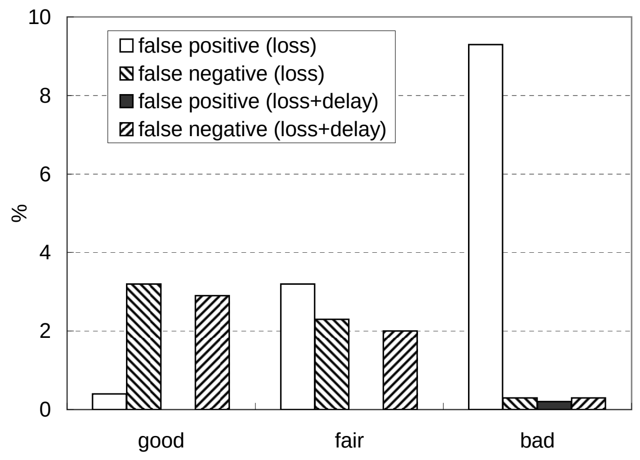

Figure 5.18: Clustering accuracy in tree merging for 30 receivers (loss)

condition in Table 5.1 have small effects on the packet delays, we plot the results in Figure 5.17 as the topology size varies. As we shown in the figure, the larger topology tends to have higher false positive and the smaller topology tends to have higher false negative. This can be explained as follows. When using delay, D_i measures the end-to-end delay discrepancies across senders. The larger the topology is, the more links it has and the larger D_i are, which makes the relative measure $D_{i,j}^{relative}$ smaller, resulting in more false positive. The same reason explains why there is more false negative for small topologies.

Figure 5.18 presents the clustering accuracy in tree merge for 30 receivers when using loss rate. We test the accuracy in three loss conditions. As shown in the figure, when loss rate is high, the clustering generates high false positive. When the loss rate is low, the clustering generates high false negative. As discussed in Section 5.3.2.B, we essentially use the logarithmic form of the success probability in the computation. When the loss rate is low, the success probability is close to one, and the performance difference D_i is close to zero which makes $D_{i,j}^{relative}$ higher, causing more false negative. Similar reason results in high false positive under high loss rate. Additionally, we also plot in the figure the clustering accuracy when combining both metric together. That is, only when $D_{i,j}^{relative}$ for two metrics both smaller than the corresponding thresholds, R_i and R_j are clustered together. As we can see, the combination of two metrics greatly reduces the number of the false positives. Reducing the number of false negatives can be achieved by first increasing the value of the threshold and then combining two metrics together.

5.5 Internet Experiments

We now describe validation of our techniques using Internet experiments.

5.5.1 Measurement Methodology

The goal of our Internet experiments is to infer the network topology from the performance measurements in terms of latency between any pairs of routers in a given ISP. Ideally, we would like to consider the loss rate as well. However, due to the rate limiting on the ICMP queries, we could not obtain enough samples to measure the loss rate accurately. To ensure the coverage, we use a distributed set of PlanetLab nodes as probing sources, each of which probe a pre-computed set of destinations. The destination assignment is carefully designed to maximize the coverage of the ISP in focus while minimizing the load on each probing hosts, which is similar to the Netdiff system [104].

In today's Internet, a large ISP has hundreds of routers, which makes the task of the analysis of the experiment data and then the inference of the topology very challenging. To reduce the complexity, we consider each PoP (Point of Presence) rather than each router as the unit of our analysis. It is justified by the fact that the performance of intra-PoP hops is negligible compared to that of inter-PoP hops. Table 5.2 presents the statistics of PoP-level topology for large ISPs. As we can see, large ISPs has a large number of PoPs. The tree topology rooted at each PoP (acting as an ingress), however, is not very complex as the average of inter-PoP hop counts (i.e., inter-PoP path length) is around 3. That is, traffic across the ISPs traverses an average 3 PoP-level hops. European ISPs tend to have a small topology size than ISPs that have major presence in US.

In the experiment, we use traceroute to probe the corresponding ingress/egress PoPs of one ISP. The probe packets are constructed with pre-computed TTL value which is expected to trigger ICMP TTL expiration response from the corresponding routers. We then obtain the route-trip time of probe packets for both ingress and egress PoPs of the ISP under test. Each hop is probed 200 times. Then we use the minimum value of all the delays for each hop to reduce the impact of measurement noise. A half of their difference is thus the delay between the ingress and the egress. We select the minimum latency as it best represents the transmission delay, which is most likely to reveal the internal physical sharing. In total we use 4 weeks of data.

ISP	AS number	# PoPs	Inter-PoP path length
Qwest	209	69	2.31
UUnet	701	147	2.67
Sprint	1239	76	4.02
AOL	1668	41	3.48
XO Comm.	2828	64	3.48
Verio	2914	58	3.46
Deutsche Telecom	3320	87	1.79
Level3	3356	84	3.58
Global Crossing	3549	73	1.96
Savvis	3561	51	2.03
France Telecom	5511	42	2.87
AT&T	7018	135	3.41
Abilene	11537	30	1.05

Table 5.2: Statistics of ISP's PoP-level topology

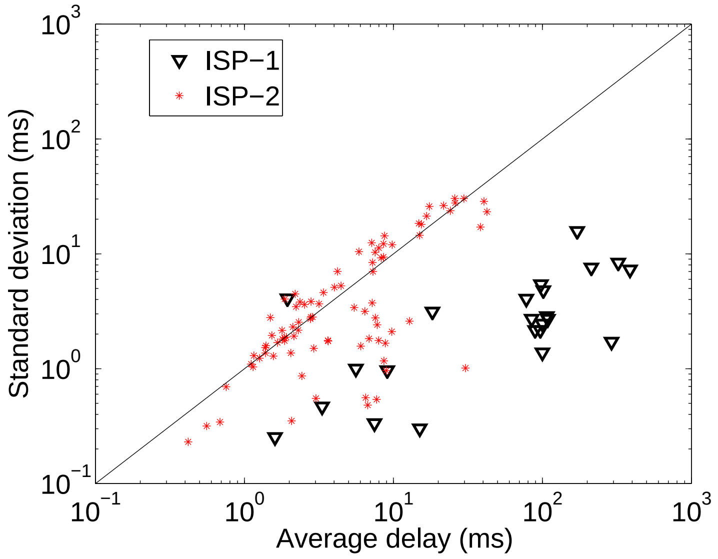

Figure 5.19: Scatter plot of inter-PoP delays

5.5.2 Experimental Results

Obviously, the performance of the tree merge algorithm relies on the accuracy of the measured delay between the ingress and the egress. Since our measurement is based on PoP level, it is necessary to examine whether the PoP-level delay exhibits large variations. Figure 5.19 plots

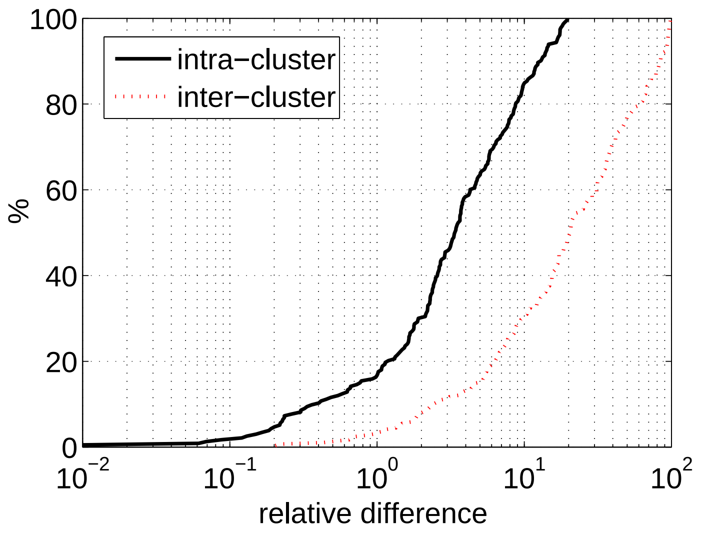

Figure 5.20: CDF of delay difference

the average delay and the corresponding delay standard deviations between any two PoPs in two typical ISPs. There are two major factors that contribute to the inter-PoP delay variations. First, due to the practice of load balancing, there are many distinct IP-level paths connecting two PoPs and these IP-level paths exhibit delay variations. Second, the variation of the queuing delay, which depends on the load of the cross traffic, also causes the delay variance. In Figure 5.19, ISP-1 has fewer PoPs and a relatively simpler network topology, while ISP-2 has a much larger and more complicated network. Inter-PoP delays in ISP-1 is shown to be much more stable than those in ISP-2. In the meantime, for long-haul inter-PoP links, *i.e.*, those with delay of hundreds of milliseconds, the standard deviation can be tens of milliseconds. Due to the relative stability of ISP-1, we use it as the focus of our experiment.

Figure 5.20 presents the cumulative distribution function of the relative difference $D_{i,j}$ for intra-cluster and inter-cluster receivers in ISP-1. Compared with Figure 5.16, it is a little more difficult to find a clear cut to distinguish intra-cluster receivers from inter-cluster receivers. When $D_{i,j}$ is less than 1% or more than 10%, we are certain of the clustering inference. However, when $D_{i,j}$ is between 1% and 10%, we cannot determine whether the receivers should be clustered. There are a few aspects we can improve in real deployment of our proposed technique. First of all, due to the limited resources, our probing source is far away from the ISP, and we are not able to directly measure the delay between the ingress and the egress of that ISP. Our current measured delay as a half of the difference of the round-trip time from the probing source to the ingress and egress, is affected by the path performance before the probing packet reaching the ISP, which explains the large variations in Figure 5.19. In practice, it is the neighboring AS that conducts the analysis as illustrated in Figure 5.1, so the delay variation will be smaller. Second, neighboring networks would also be able to obtain other performance measurements, *e.g.*, loss or delay jitter, which could help determine the clustering, as described in Section 5.3.2. Third, in Section 5.3.2.C, we describe that the clustering results can be validated with the separate tree structures to improve the accuracy of the clustering.

5.6 Related Work

There are several measurement tools related to our study. For example, the mtrace [107] tool, viewed as a counterpart of traceroute in multicast, acquires the route from a multicast source to a receiver, along with other information (*e.g.*, per-hop loss statistics) about the path by using an MTRACE tracing feature implemented in multicast routers that is accessed as an extension to the IGMP protocol. The Tracer tool [108] uses mtrace to organize the receivers deterministically into a logical tree structure to achieve effective error recovery and congestion control. These tools rely on the responsiveness of the routers to measurement queries and its effectiveness is thus limited.

In the context of network tomography, Ratnasamy and McCanne [103] and Caceres *et al.* [109] first demonstrated that the correlations of the end-to-end loss measurements in a multicast session could be used to identify the multicast tree topology. Several studies [99, 102] have followed to rigorously establish the correctness of these techniques and developed a general framework in which other measurements, such as delay, delay variance, can be used. This body of work has been further extended to unicast scenarios as in [97, 98, 100, 110]. For example, the striped unicast probing proposed in [100] uses a sequence of back-to-back packets sent to different receivers as an approximation of a multicast probe, thus enabling them to use link loss and delay inference techniques developed for multicast probes. Similarly, Coates *et al.* [98] proposed a scheme called “sandwich probing” in which two small probing packets to one receiver is sandwiched by another much larger packet destined for the other receiver. The objective is to ensure the second small packet queued behind the large packet captures the delay characteristics of the shared path between the two receivers. Authors of [111, 112] provide a survey of the related work.

Our study uses a deterministic classification algorithm that recursively aggregate receivers together to generate a binary tree in a bottom-up fashion. It has been shown [98] that such greedy nature of the deterministic algorithm based on local decisions (the correlation between a pair of nodes) can lead to suboptimal results. To avoid the pitfall, [98] proposed another method based on the idea of sampling, called "Markov-chain Monte Carlo" (MCMC) approach. The MCMC procedure searches through the topology space (*i.e.*, the set of all possible tree structures) and selects the one with achieve maximum likelihood estimation from the end-to-end measurements. The advantages of MCMC is that it identifies the topology globally rather than incrementally decides a small piece of the tree at a time. Because the tree construction is not the focus of our study, we only investigate the performance of the deterministic clustering algorithm which has been demonstrated to have comparable inference accuracy with other approaches [98].

In [113], Coates *et al.* proposed a merging technique that is motivated by a similar observation as ours. However, our study adopts different tree merge heuristics to identify the complete topology more accurately. In addition, we also investigate how this technique can be applied in the real Internet environment, extensively evaluating the effectiveness of our algorithm.

5.7 Concluding Remarks

In this chapter, we proposed a framework to discover the resource sharing of a network with the end-to-end measurements between the ingress and egress of the network. The tree merge technique aggregates the tree topologies associated with different senders by comparing across the set of receivers with their performance discrepancies from the senders, and thus can be a good complement to the previous tree construction algorithms to identify network topologies. Moreover, the tree merge procedure can also be used to verify the correctness of the constructed tree and help improve the accuracy of the tree inference. Our work presents an important step towards ensuring high network performance without network cooperations despite increasingly opaque IP networks.

Through extensive simulations and measured experiments, we demonstrated the utility and accuracy of the proposed merge algorithm. By combining both the loss rate and the delay metrics, it correctly clusters receivers in more than 99% of the tests. Such insight is critical for improving network failure resilience given today’s increasingly opaque IP networks.

CHAPTER 6
Conclusions and Future Work

This dissertation has focused on how to enhance the robustness of the current Internet interdomain routing. In particular, our approach is taken into two directions. In one direction, we have proposed techniques to assist in pinpointing the origin and cause of the routing instability for a single network *after* to failures occurred. In the other direction, based on the characterization of today's Internet interdomain routing resilience, we have proposed *proactive* techniques that can equip the routing system with the capability to tolerate certain types of Internet failures. This dissertation makes several contributions toward the understanding of the Internet interdomain routing (e.g., the routing dynamics and the effect of policy-driven routing), which have potential for making a significant impact on the design of the next-generation Internet routing infrastructure.

We summarize the primary contributions of this dissertation, which is followed by a discussion of future research directions.

6.1 Primary Contributions

This dissertation makes the following contributions toward improving the robustness of Internet interdomain routing.

- Routing dynamics have always been a major concern of the Internet engineering community because it not only incurs bandwidth and processing overhead on routers, but may also lead to poor end-to-end performance (e.g., packet losses, delay, delay jitter). Understanding routing dynamics allow us to pinpoint network anomalies and pathologies, identify potential protocol or router design defects, and suggest better designs of future routing infrastructures. There is a large body of literature analyzing BGP messages to identify the locations and causes of routing changes. Our work, however, differs from them in that it focuses on organizing a large number of BGP updates seen in a *single* network and develop a network troubleshooting system that can capture several types of routing anomalies in real-time. Applying the system to a large ISP, we have several surprising findings that can help network operators improve the routing stability of their network. For example, despite having route-flap damping features enabled on all of the routers, our tool identified a number of persistently flapping prefixes that are caused by other routing instability factors. We also found that hot-potato routing changes and eBGP session resets were responsible for many of the large routing disruptions.

- While the Internet becomes more and more ubiquitous in every aspect of our lives, we have been constantly witnessing network outages caused by accidental cable cuts, hardware malfunctions, power outage, human errors, natural disasters, or even terrorist attacks and DoS attacks. It has been reported that since 1992 there have been about 16 outages per month in the United States alone that each affected 30,000 users. The robustness of the Internet routing is thus critical under these extreme conditions, failures such as 911 attacks, 2003 Northeast Blackout, and 2006 Taiwan Earthquake. We have proposed a framework to systematically analyze how the current Internet interdomain routing system reacts to various types of failures by establishing a realistic failure model, and then pinpoints reliability bottlenecks of the Internet. Our main results on the characteristics of BGP routing resilience in face of large-scale failures are summarized as follows. First, Tier-1 depeering, despite its infrequent occurrence, disrupts 94% of the reachability between the single-homed customer ASes of the affected Tier-1 ASes. Second, most of the reachability damage in today's Internet is caused by failures of the critical access links, which are traversed by all possible paths from the affected AS to the rest of the Internet. 32% of the ASes are vulnerable to this type of failure. Third, the BGP policy limits the ASes' option in selecting paths to reach other ASes, an additional 6% non-stub ASes can be disrupted by a single access link failure even though the physical connectivity might be available to bypass the failure.

- The current Internet interdomain routing is policy-driven, meaning that the physical connectivity does not imply reachability. As shown in our work on characterization of BGP's resilience to large-scale failures, the policy limits the ASes' option in selecting redundant physical connectivity to bypass failures. Thus, we have proposed a novel idea of *Dynamic Routing Negotiation* (DRN) to allow ISPs to temporarily relax policy restrictions when needed, to enhance the Internet's routing robustness by exploiting the existing physical redundancy in the network topology. By simulating failure scenarios on a realistic Internet topology, we evaluate DRN for two common types of failures that often disrupt reachability and demonstrate that policy relaxation on peering links alone can reduce the percentage of AS pairs disconnected by failures, down to 0% and 37%, respectively. DRN can make further resilience enhancements with other types of policy relaxation.

- The knowledge of network topology can always be beneficial to diagnosis of network anomalies, such as link failures and congestion, and devising measures to alleviate their effects. Unfortunately, due to the increasing concerns on network security, compounded by the recent emergence of MPLS-like layer-2 technology, traditional topology measurement tools such as traceroute becomes less capable of identifying the internal structure of networks. To reduce the opaque nature of today's networks, we propose a novel approach to discovering the internal structure of each network based on the performance measurement obtained between each pair of ingress and egress points in the network. In particular, we develop a "tree-merge" technique to consolidate together trees that inferred across different ingress points. We use extensive simulations and real Internet experiments to demonstrate the utility and accuracy of our algorithm. This makes an important contribution in enhancing network reliability by improving the transparency of today's increasingly opaque networks from the perspective of shared resources.

6.2 Future Work

Our work on enhancing the robustness of the Internet interdomain routing can be extended further in the following directions:

- **Automation of the response to routing disruptions:** In Chapter 2, our troubleshooting tool generates a report that concisely identifies the set of important events (e.g., large routing disruptions, persistently flapping prefixes) that warrant network operators' attention. More work needs to be done to automate the process of responding to the reported routing events and reduce the involvement of human intervention. We consider the effects of routing configuration changes on different types of routing disruptions. For example, we can adjust route-flap damping parameters to further penalize persistently-flapping prefixes or use static route to prevent routes switch back and forth between several possible options.

- **Analysis on external routing instability:** In Chapter 2, we focused on events that cause the largest routing disruptions to the network, which happen to be due to either internal hot-potato changes or eBGP session resets. In future, we want to delve into more detail of category "multiple external disruptions" to obtain a better understanding of its routing symptom, such as how to pinpoint the accountable AS that originally causes the instability. We also plan to explore routing architectures, operational practices, and protocol enhancements that reduce the likelihood and impact of the routing disruptions associated with hot-potato changes and eBGP session resets.

- **Construction of a more accurate Internet topology map:** In Chapters 3 and 4, the accuracy of our analysis results hinges on the completeness of the topology map and the accuracy of the AS relationship. Certain links, especially peer-to-peer links in the edge of the Internet, only appear in the BGP paths between their associated ASes, and therefore cannot be captured by a limited number of vantage points, such as RouteViews and RIPE. Moreover, the AS relationship is inferred based on various types of simple heuristics and cannot capture the complicated relationship among ASes. Even worse, because the inference of a relationship is highly dependent on the inference of other relationships, one mistake in the process can have cascading effects on the accuracy of the whole topology map. In future, we will consider combining multiple inference heuristics to obtain a more accurate view of the Internet.

- **Obtaining an AS-level traffic matrix:** In Chapter 3, we simplify the evaluation of the traffic impacts of each failure by using the number of paths that traverse each AS link as the traffic flowing through that link. In future, we would like to develop techniques to accurately estimate the traffic distribution matrix across ASes and then incorporate into our current analysis to have a better understanding of routing impacts.

- **Applicability of the topology inference technique:** In Chapter 5, we discuss the challenges in applying the topology inference technique to the real Internet. For example, ISPs apply a variety of load-balancing techniques to packets through their networks, and the topology seen by a sequence of probing packets might change on the fly, thus introducing inaccuracies into our analysis. In addition, the queuing delay might be a more dominating factor than the propagation delay, which is the foundation of our tree-merge algorithm. Therefore, our analysis needs to find ways to eliminate the effects of the queuing delay on the measurement. All of this requires us to gain more experience from the deployment of our tool at different types of ISPs to see whether and how these factors affect the accuracies of our analysis.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] "Developing a next-generation Internet architecture." `http://www.isi.edu/newarch/DOCUMENTS/WhitePaper.ps`.

[2] D. Clark, "The design philosophy of the DARPA Internet protocol," in *Proc. ACM SIGCOMM*, August 1988.

[3] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)." Internet Draft draft-ietf-idr-bgp4-26.txt, work in progress, October 2004.

[4] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and Interdomain routing," *IEEE/ACM Trans. Networking*, vol. 10, no. 2, pp. 232–243, 2002.

[5] N. Feamster, H. Balakrishnan, and J. Rexford, "Some foundational problems in interdomain routing," in *In ACM SIGCOMM Workshop on Hot Topics in Networking (HOTNets-III*, pp. 41–46, 2004.

[6] C. Labovitz, R. Malan, and F. Jahanian, "Internet routing instability," in *Proc. ACM SIGCOMM*, 1997.

[7] C. Labovitz, R. Malan, and F. Jahanian, "Origins of internet routing instability," in *Proc. IEEE INFOCOM*, 1999.

[8] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence," *IEEE/ACM Trans. on Networking*, vol. 9, pp. 293–306, June 2001.

[9] T. Griffin, "What is the sound of one route flapping?," in *IPAM*, 2002.

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

[11] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, “Locating Internet routing instabilities,” in *Proc. ACM SIGCOMM*, August 2004.

[12] D.-F. Chang, R. Govindan, and J. Heidemann, “The temporal and topological characteristics of BGP path changes,” in *Proc. ICNP*, November 2003.

[13] T. Wong, V. Jacobson, and C. Alaettinoglu, “Making sense of BGP,” February 2004. NANOG presentation.

[14] M. Lad, A. Nanavati, D. Massey, and L. Zhang, “An algorithmic approach to identifying link failures,” in *Proc. Pacific Rim Dependable Computing*, 2004.

[15] A. P. Snow and M. W. Thayer, “Defeating telecommunication system fault-tolerant design,” in *Proc. ISW 2000*, October 2000.

[16] “Anchor-Draggers Cut Asia’s Internet Pipe.” `http://news.zdnet.co.uk/internet/0,1000000097,2095715,00.htm`.

[17] “Cable Failure Hits UK Internet Traffic.” `http://news.zdnet.co.uk/0,39020330,39118125,00.htm`.

[18] “Ruptured Cable Disrupts Internet Service.” `http://www.msnbc.msn.com/id/19363112/`.

[19] “Internet Routing Behavior on 9/11.” `http://www.renesys.com`.

[20] “Blackouts Cause N America Chaos.” `http://news.bbc.co.uk/1/hi/world/americas/3152451.stm`.

[21] “Asia scrambles to restore communications after quake.” `http://www.iht.com/articles/2006/12/28`.

[22] R. Teixeira and J. Rexford, “A measurement framework for pin-pointing routing changes,” in *Proc. SIGCOMM Network Troubleshooting Workshop*, September 2004.

[23] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, “Observation and analysis of BGP behavior under stress,” in *Proc. Internet Measurement Workshop*, 2002.

[24] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, “BGP routing stability of popular destinations,” in *Proc. Internet Measurement Workshop*, 2002.

[25] “Netflow.” `http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml`.

[26] N. Duffield, C. Lund, and M. Thorup, “Estimating flow distributions from sampled flow statistics,” in *Proc. ACM SIGCOMM*, 2003.

[27] O. Maennel and A. Feldmann, “Realistic BGP traffic for test labs,” in *Proc. ACM SIGCOMM*, 2002.

[28] Z. M. Mao, R. Bush, T. G. Griffin, and M. Roughan, “BGP beacons,” in *Proc. Internet Measurement Conference*, 2003.

[29] C. Villamizar, R. Chandra, and R. Govindan, “BGP route flap damping,” *RFC 2439*, 1998.

[30] C. Lonvick, “The BSD syslog protocol.” RFC 3164, August 2001.

[31] T. Griffin and G. Wilfong, “Analysis of the MED Oscillation problem in BGP,” in *Proc. IEEE ICNP*, November 2002.

[32] D. McPherson, V. Gill, D. Walton, and A. Retana, “Border gateway protocol (BGP) persistent route oscillation condition.” RFC 3345, August 2002.

[33] E. Chen and T. Bates, “An application of the BGP community attribute in multi-home routing,” *RFC 1998*, 1996.

[34] “RIPE Routing-WG recommendations for coordinated route-flap damping parameters,” October 2001. Document ID: ripe-229, `http://www.ripe.net/ripe/docs/routeflap-damping.html`.

[35] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, “Dynamics of hot-potato routing in IP networks,” in *Proc. ACM SIGMETRICS*, 2004.

[36] N. Feamster, Z. M. Mao, and J. Rexford, “BorderGuard: Detecting cold potatoes from peers,” in *Proc. Internet Measurement Conference*, October 2004.

[37] D. Chang, R. Govindan, and J. Heidemann, “An empirical study of router response to large BGP routing table load,” in *Proc. Internet Measurement Workshop*, November 2002.

[38] “University of Oregon Route Views Archive Project.” `http://www.routeview.org`.

[39] R. Teixeira, N. Duffield, J. Rexford, and M. Roughan, “Traffic matrix reloaded: Impact of routing changes,” in *Proc. Passive and Active Measurement Workshop*, March/April 2005.

[40] W. Fang and L. Peterson, “Inter-AS traffic patterns and their implications,” in *Proc. IEEE Global Internet*, December 1999.

[41] R. Mahajan, D. Wetherall, and T. Anderson, “Understanding BGP misconfiguration,” in *Proc. ACM SIGCOMM*, August 2002.

[42] M. Roughan, T. Griffin, Z. M. Mao, A. Greenberg, and B. Freeman, “Combining routing and traffic data for detection of IP forwarding anomalies,” in *Proc. SIGCOMM Network Troubleshooting Workshop*, 2004.

[43] “RIS Raw Data.” `http://www.ripe.net/projects/ris/rawdata.html`.

[44] “Public route servers.” `http://www.bgp4.net`.

[45] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz, “Characterizing the Internet hierarchy from multiple vantage points,” in *Proc. IEEE INFOCOM*, 2002.

[46] R. Cohen and D. Raz, “The Internet dark matter - on the missing links in the as connectivity map,” in *Proc. IEEE INFOCOM*, April 2006.

[47] Y. He, G. Siganos, M. Faloutsos, and S. Krishnamurthy, “A systematic framework for unearthing the missing links: Measurements and impact,” in *Proc. NSDI*, April 2007.

[48] “CAIDA AS Relationships.” `http://as-rank.caida.org/data/`.

[49] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, kc claffy, and G. Riley, “AS Relationships: Inference and Validation,” *ACM Computer Communication Review*, vol. 37, no. 1, 2007.

[50] J. Xia and L. Gao, “On the Evaluation of AS Relationship Inferences,” in *Proc. IEEE Global Internet*, 2000.

[51] L. Gao, “On Inferring Autonomous System Relationships in the Internet,” in *Proc. IEEE Global Internet*, 2000.

[52] Z. M. Mao, L. Qiu, J. Wang, and Y. Zhang, “On AS-Level Path Inference,” in *Proc. ACM SIGMETRICS*, 2005.

[53] G. Battista, M. Patrignani, and M. Pizzonia, “Computing the Types of the Relationships Between Autonomous Systems,” in *Proc. IEEE INFOCOM*, March 2003.

[54] W. Muehlbauer, S. Uhlig, B. Fu, M. Meulle, and O. Maennel, “In search for an appropriate granularity to model routing policy,” in *Proc. ACM SIGCOMM*, August 2007.

[55] X. Dimitropoulos and G. Riley, “Modeling Autonomous System Relationships,” in *Proceeding of 20th Principles of Advanced and Distributed Simulation (PADS)*, 2006.

[56] L. Gao, T. G. Griffin, and J. Rexford, “Inherently safe backup routing with BGP,” in *Proc. IEEE INFOCOM*, 2001.

[57] “Internet Routing Resilience Project Page.” `http://www.eecs.umich.edu/~wujz/irrf/`.

[58] S. M. Bellovin and E. R. Gansner, “Using Link Cuts to Attack Internet Routing.” May 2003.

[59] J. Wu, Z. M. Mao, J. Rexford, and J. Wang, “Finding a needle in a haystack: Pinpointing significant bgp routing changes in an ip network,” in *Proc. NSDI*, May 2005.

[60] “The backhoe: A real cyberthreat.” `http://www.wired.com/science/discoveries/news/2006/01/70040`.

[61] “ISP spat blacks out Net connections.” `http://www.networkworld.com`.

[62] “NANOG mailing list.” `http://www.merit.edu/mail.archives/nanog/`.

[63] “UUnet backbone problems slow down the Net.” `http://www.itworld.com`.

[64] “Impact of Hurricane Katrina on Internet infrastructure.” `http://www.renesys.com`.

[65] “PlanetLab.” `http://www.planet-lab.org`.

[66] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, 2001.

[67] "NetGeo - The Internet Geographic Database." `http://www.caida.org/tools/utilities/netgeo/index.xml`.

[68] R. Albert, H. Jeong, and A.-L. Barabasi, "Error and attack tolerance of complex networks," *Nature*, 2000.

[69] R. Cohen, K. Erez1, D. ben Avraham, and S. Havlin, "Resilience of the Internet to Random Breakdowns," *Phys. Rev. Lett.*, 2000.

[70] X. Zhao, B. Zhang, A. Terzis, D. Massey, and L. Zhang, "The Impact of Link Failure Location on Routing Dynamics: A Formal Analysis,," in *Proceedings of ACM SIGCOMM Asia Workshop*, 2005.

[71] W. Muhlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig, "Building an AS-Topology Model," in *Proc. of ACM SIGCOMM*, 2006.

[72] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker, "Characterizing and measuring path diversity of internet topologies," in *Proc. ACM SIGMETRICS*, September 2003.

[73] T. Erlebach, A. Hall, L. Moonen, A. Panconesi, F. Spieksma, and D. Vukadinovic, "Robustness of the Internet at the Topology and Routing Level," *Lecture Notes in Computer Science*, vol. 4028, 2006.

[74] H. Wang, Y. R. Yang, P. H. Liu, J. Wang, A. Gerber, and A. Greenberg, "Reliability as an interdomain service," in *Proc. ACM SIGCOMM*, August 2007.

[75] "Middle East and Asia lose Internet access after cable fails." `http://www.guardian.co.uk/technology/2008/jan/30/asia.internet.outage`.

[76] X. Yang, D. Wetherall, and T. Anderson, "Source selectable path diversity via routing deflection," in *Proc. ACM SIGCOMM*, August 2006.

[77] M. Motiwala, N. Feamster, and S. Vempala, "Better interdomain path diversity with BGP path splicing," in *PRESTO*, 2007.

[78] W. Xu and J. Rexford, "MIRO: Multi-path interdomain routing," in *Proc. ACM SIGCOMM*, August 2006.

[79] N. Kushman, S. Kandula, D. Katabi, and B. M. Maggs, "R-BGP: Staying connected in a connected world," in *Proc. NSDI*, April 2007.

[80] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," *RFC 1771*, 1995.

[81] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, "Achieving convergence-free routing using failure-carrying packets," in *Proc. ACM SIGCOMM*, August 2007.

[82] S. Lee, Y. Yu, S. Nelakuditi, Z. Zhang, and C.-N. Chuah, "Proactive vs reactive approaches to failure resilient routing," in *Proc. IEEE INFOCOM*, March 2004.

[83] H. Wang, Y. R. Yang, P. H. Liu, J. Wang, A. Gerber, and A. Greenberg, "Reliability as an interdomain service," in *Proc. ACM SIGCOMM*, August 2007.

[84] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. OSDI*, December 2001.

[85] A. Broido and kc claffy, "Analysis of RouteViews BGP data: policy atoms," in *Proc. of NRDM workshop*, May 2001.

[86] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Trans. Networking*, vol. 9, no. 6, pp. 681–692, 2001.

[87] C. T. Ee, V. Ramachandran, B.-G. Chun, K. Lakshminarayanan, and S. Shenker, "Resolving Inter-domain policy disputes," in *Proc. ACM SIGCOMM*, August 2007.

[88] G. Maier, A. Pattavina, S. D. Patre, and M. Martinelli, "Optical network survivability: Protection techniques in the WDM layer," *Photonic Network Communications*, vol. 4, July 2002.

[89] K. Murakami and H. S. Kim, "Virtual path routing for survivable ATM networks," *IEEE/ACM Trans. Networking*, vol. 4, February 1996.

[90] A. Fumagalli and L. Valcarenghi, "IP restoration vs. WDM protection: Is there an optimal choice?," *IEEE Network Magazine*, vol. 14, November 2000.

[91] A. Bremler-Barr, Y. Afek, and S. Schwarz, “Improved BGP convergence via ghost flushing,” in *Proc. IEEE INFOCOM*, April 2003.

[92] W. Sun, Z. M. Mao, and K. G. Shin, “Differentiated BGP update processing for improved routing convergence,” in *Proc. ICNP*, October 2006.

[93] D. Pei, M. Azuma, D. Massey, and L. Zhang, “BGP-RCN: Improving BGP convergence through root cause notification,” *Computer Networks and ISDN Systems*, vol. 48, June 2005.

[94] R. Mahajan, D. Wetherall, and T. Anderson, “Negotiation-based routing between neighboring ISPs,” in *Proc. NSDI*, April 2005.

[95] R. Mahajan, D. Wetherall, and T. Anderson, “Mutually controlled routing with independent ISPs,” in *Proc. NSDI*, April 2007.

[96] N. Spring, R. Mahajan, and D. Wetherall, “ISP topologies with Rocketfuel,” in *Proc. ACM SIGCOMM*, 2002.

[97] M. Coates and R. Nowak, “Network loss inference using unicast end-to-end measurement,” in *Proc. ITC Conf. IP Traffic, Modelling and Management*, September 2000.

[98] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang, “Maximum likelihood network topology identification from edge-based unicast measurements,” in *Proc. ACM SIGMETRICS*, 2002.

[99] R. Castro, M. Coates, and R. Nowak, “Likelihood based hierarchical clustering,” *IEEE Trans. Signal Processing*, vol. 52, pp. 2308–2321, August 2004.

[100] N. G. Duffield, F. L. Presti, V. Paxson, and D. F. Towsley, “Inferring link loss using striped unicast probes,” in *Proc. IEEE INFOCOM*, 2001.

[101] N. G. Duffield, J. Horowitz, and F. L. Presti, “Adaptive multicast topology inference,” in *Proc. IEEE INFOCOM*, 2001.

[102] N. G. Duffield, J. Horowitz, F. L. Presti, and D. F. Towsley, “Multicast topology inference from measured end-to-end loss,” *IEEE Trans. Inform. Theory*, vol. 48, pp. 26–45, January 2002.

[103] S. Ratnasamy and S. McCanne, “Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements,” in *Proc. IEEE INFOCOM*, 1999.

[104] R. Mahajan, M. Zhang, L. Poole, and V. Pai, “Uncovering Performance Differences in Backbone ISPs with Netdiff,” in *Proceeding of NSDI*, 2008.

[105] B. Augustin, T. Friedman, and R. Teixeira, “Measuring load-balanced paths in the internet,” in *IMC ’07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 149–160, 2007.

[106] “NS2 Network Simulator.” `http://www.isi.edu/nsnam/ns/`.

[107] “mtrace: Probe Multicast Path from a Source to a Receiver.” `ftp://ftp.parc.xerox.com/pub/net-research/ipmulti`.

[108] B. N. Levine, S. Paul, and J. J. Garcia-Luna-Aceves, “Organizing multicast receivers deterministically by packet-loss correlation,” *Multimedia Systems*, vol. 9, no. 1, pp. 3–14, 2003.

[109] R. Caceres, N. G. Duffield, J. Horowitz, D. Towsley, and T. Bu, “Multicast-based inference of network internal loss characteristics: Accuracy of packet estimation,” in *Proc. IEEE INFOCOM*, 1999.

[110] A. Bestavros, J. Byers, and K. Harfoush, “Inference and labeling of metric-induced network topologies,” in *Proc. IEEE INFOCOM*, 2002.

[111] M. Coates, A. O. H. III, R. Nowak, and B. Yu, “Internet tomography,” *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47–65, 2002.

[112] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, “Network tomography: Recent developments,” *Statistical Science*, vol. 19, no. 3, pp. 499–517, 2004.

[113] M. Coates, M. Rabbat, and R. Nowak, “Merging logical topologies using end-to-end measurements,” in *Proc. of IMC*, 2003.