The Pennsylvania State University
The Graduate School

DETECTING AND RECOVERING FROM LARGE-SCALE FAILURES IN THE INTERNET

A Dissertation in
Computer Science and Engineering
by
Qiang Zheng

© 2012 Qiang Zheng

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2012

The dissertation of Qiang Zheng was reviewed and approved* by the following:

Guohong Cao
Professor of Computer Science and Engineering
Dissertation Advisor, Chair of Committee

Thomas F. La Porta
William E. Leonhard Professor of Computer Science and Engineering

Sencun Zhu
Associate Professor of Computer Science and Engineering

Tao Yao
Assistant Professor of Industrial and Manufacturing Engineering

Raj Acharya
Professor of Computer Science and Engineering
Department Head of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

The Internet has become a significant and widely used infrastructure for a wide range of communication and other services. Internet reliability and availability are crucial to many applications such as financial transactions, online games, Voice over IP, and video services. However, large-scale failures, caused by events like natural disasters and intentional attacks, are common in the Internet. Unlike sporadic and isolated link failures, large-scale failures usually lead to serious routing disruption and severe packet loss. Therefore, efficiently monitoring the Internet to detect large-scale failures and quickly recovering from failures are particularly important for enhancing Internet reliability and availability.

The specific goal of this dissertation is to provide comprehensive solutions for detecting large-scale failures in the Internet and quickly recovering from large-scale failures. First, we design an approach for monitoring link performance using tomography-based end-to-end probe. Given a set of links to monitor, the objective is to select the minimum number of probing paths to uniquely determine the performance of all identifiable links and cover all unidentifiable links. Through this, we can identify the links with abnormal performance. Second, we propose a two-phase approach for detecting and localizing large-scale router failures using traceroute-like active probes. To detect large-scale router failures, the detection phase is periodically invoked to probe all routers. When detecting large-scale router failures, the localization phase is triggered to identify the failed routers. We reduce the probing cost by avoiding three types of useless probes. For the routers whose status cannot be identified by probes, we develop a distance-based method to estimate their failure probability. Third, we design an approach called Reactive Two-phase Rerouting (RTR) for intra-domain routing to quickly recover from geographically correlated failures with the shortest recovery paths. To recover a failed routing path, RTR first forwards packets around the failure area to collect information on failures. Then, in the second phase, RTR calculates a

new shortest path and forwards packets along it through source routing. RTR can deal with geographically correlated failures associated with areas of any shape and location. Finally, we propose two cross-layer approaches built on backup paths to recover intra-domain routing from correlated link failures. Based on the mapping between the IP layer and optical layer topologies, we develop a correlated failure probability (CFP) model and a probabilistically correlated failure (PCF) model to quantify the impact of IP link failure on the reliability of backup paths. With the CFP model, we propose two algorithms to select a backup path to protect each IP link. The first algorithm aims at choosing the backup paths with minimum failure probability, and the second algorithm further considers the bandwidth constraint and aims at minimizing the traffic disruption caused by failures. With the PCF model, we propose an algorithm to choose multiple reliable backup paths for each IP link to minimize the routing disruption caused by failures.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank all people who have helped and inspired me during my doctoral study.

I especially would like to express my sincere gratitude to my advisor Prof. Guohong Cao for his consistent support and supervision. I am grateful for the considerable time he had spent on discussing our research problems and training me to be a matured researcher. He is always there whenever I need guidance or help. Without his encouragement and great efforts, I could not have completed my Ph.D study.

I wish to express my warm and sincere thanks to Prof. Thomas F. La Porta for the inspirational and insightful comments on my papers, which ensured I was on the right track towards finishing my dissertation. I also want to thank Prof. Sencun Zhu and Prof. Tao Yao whose thoughtful advices often gave me a sense of direction during my Ph.D. studies. It is my great honor to have these wonderful professors in my doctoral committee. Without their generous help and encouragement, the dissertation could not have reached the present form.

My gratitude also goes to my co-author Dr. Ananthram Swami in Army Research Laboratory for his inspiring thoughts and insightful suggestions for helping me improve the qualify of our papers. I would like to thank another co-author Jing Zhao for his hard work on our paper published in IEEE/IFIP DSN 2012. I also want to thank my labmates Bo Zhao, Wei Gao, and Qinghua Li for the encouragement on my work and for making my years of study at Penn State University an enjoyable and memorable journey.

Last but not least, I am deeply indebted to my parents Guofu Zheng and Guiling Liu for their unconditional love. They have always supported and encouraged me to do my best in all matters of my life. I also wish to express my deepest gratitude to my wife Jin Huang for her constant faith in me and sacrifices made during my candidature. This dissertation is simply impossible without them.

Dedication

To my parents and my wife.

Chapter 1

Introduction

The Internet has rapidly grown in the past decade. It spreads to many countries and consists of thousands of Autonomous Systems. A large number of applications are deployed on the Internet and used by hundreds of millions of people around the world. Many applications, e.g., financial transactions, online games, Voice over IP, and video services, are sensitive to network performance and require the Internet to have high reliability and availability.

Recent measurements [1, 2] discovered that failures are fairly common in the Internet for various reasons like router hardware and software failures, network maintenance activities, protocol misconfigurations [3], natural disasters [4, 5, 6, 7, 8], and intentional attacks [9, 10, 11]. When failures occur, a large number of packets may be dropped due to invalid routes. For example, disconnection of an OC-192 link (10 Gb/s) for 10 seconds can lead to about 12 million packets being dropped on a link (assuming that the average packet size is 1,000 bytes). Under large-scale failures, packet loss would be much more severe [12, 13].

Accurately detecting failures and quickly recovering from failures are critical to enhancing Internet reliability and availability. The routing protocols used in the Internet like OSPF [14], IS-IS [15], and BGP [16] can adapt to failures through their convergence mechanisms. However, the protocol convergence is a time consuming process, which usually takes several seconds even for a single link failure [17]. To reduce routing disruption, recovery mechanisms are used to reroute traffic affected by failures during protocol convergence. In addition to recovery, it is important to accurately identify the failed links and routers, because Internet Service Providers

need to manually replace the failed components in many cases. These demands motivate recent research on failure detection and recovery for the Internet.

1.1 Motivation

The Internet has two types of large-scale failures. The first type is *geographically correlated failures*, i.e., routers and links within a geographically contiguous area fail simultaneously. Many events like natural disasters and intentional attacks can cause geographically correlated failures. For example, Hurricane Katrina [5], the Taiwan earthquake in December 2006 [6], and the Wenchuan earthquake in May 2008 [8] destroyed a large portion of the Internet near the disaster location and resulted in serious routing disruption. Additionally, intentional attacks like terroristic events (e.g., 911 attack [9]) and attacks by weapons of mass destruction [11] can also lead to geographically correlated failures. The second type is *correlated link failures* which are caused by link cuts or attacks [10]. Internet backbone networks are primarily built on the Wavelength Division Multiplexing (WDM) infrastructure [18]. In this layered structure, the IP layer topology is embedded on the optical layer topology. An IP link consists of multiple fiber links and a fiber link is shared by multiple IP links. When a fiber link fails, all the IP links embedded on it fail simultaneously.

Extensive research works have focused on failure detection [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31] and recovery [32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43] for the Internet. However, most of them are built on an implicit assumption that there are only sporadic and isolated link failures in the Internet. Hence, they are not suitable for large-scale failures.

Prior failure detection approaches are mainly built on tomography-based end-to-end probe and reply-based probe like `ping` and `traceroute`. End systems are connected to the network and send probing message to check the status of links and routers. Based on the probing result, failure detection approaches identify the failed links and routers. Existing solutions assume that there is only single failure, or focus on finding the smallest set of failures to match the probing result. For large-scale failures, the detection accuracy of existing methods may be quite low, and many failed links and routers cannot be identified.

Most of the current recovery approaches use precomputed backup paths to reroute traffic affected by failures. However, they suffer from two limitations. First, existing approaches assume that a network only has few link failures, and thus backup paths probably have no failures. In fact, large-scale failures may destroy many links and routers. As a result, a link and its backup path may fail simultaneously. Second, prior recovery solutions keep rerouting all packets affected by failures, and do not differentiate the packets with reachable destinations from packets with unreachable destinations. Under small-scale failures, it is unlikely that there are many unreachable destinations, and hence discarding packets late has little negative impact. However, large-scale failures may make many destinations unreachable, especially when the network is partitioned by failures. Keeping forwarding packets with unreachable destinations wastes network resources and de-grades network performance.

1.2 Challenges

There are two major challenges in detecting large-scale failures as follows.

- **Maximizing the failure detection accuracy:** Most failure detection mechanisms use probing messages to detect the status of links and routers. To maximize the detection accuracy, probing paths should be carefully selected to collect as much failure information as possible. Under large-scale failures, probing messages may not reach some links and routers. Therefore, new solutions are needed to estimate the status of these links and routers.

- **Minimizing the failure detection cost:** The Internet consists of a large number of links and routers. To have better scalability and less negative impact on the normal data transmission in the Internet, failure detection mechanisms should minimize the cost of failure detection, such as using minimum number of probing paths, probing messages, or end systems for sending probing messages.

Recovering from large-scale failures has the following three major challenges.

- **Collecting failure information:** A router monitors the connection with its neighbors. When large-scale failures occur, no individual router has the overall information of failures, and each router only knows whether its neighbors are reachable. For an unreachable neighbor, the router cannot differentiate whether the neighbor fails or the link connecting the neighbor fails. Therefore, recovery mechanisms need to quickly and efficiently collect failure information, and then find suitable recovery paths to reroute traffic.

- **Selecting the best recovery paths:** IP links have different reliability and capacity, and thus recovery paths built on them have different properties. As a result, there are several considerations in recovery path selection, such as reliability, usable bandwidth, and path length. To achieve the best protection, recovery mechanisms need to select the best recovery paths and fully utilize network resources.

- **Reducing computational overhead:** Since routers calculate recovery paths, the algorithm should be simple and have low computational overhead. Complex algorithms consume much computational resources of the routers, and thus affect normal packet forwarding.

1.3 Focus of This Dissertation

The goal of this dissertation is to provide comprehensive solutions for accurately detecting large-scale failures in the Internet and quickly recovering from them. Specifically, we focus on four important aspects, i.e., network link monitoring, detecting and localizing large-scale router failures, recovery from geographically correlated failures, and recovery from correlated link failures. We briefly explain them in the following four subsections.

1.3.1 Probe-Based Network Link Monitoring

Continuously monitoring link performance is important for detecting large-scale link failures. Due to various advantages, tomography-based end-to-end probe has received much attention and has been widely used in network monitoring. In this

approach, end systems are connected to the network to send and receive probes. Each end-to-end probe measures the performance of the entire path. Generally, network links are classified into two types. If the performance of a link can be uniquely inferred by a set of probes, this link is identifiable and the set of probes can uniquely determine it. Otherwise, the link is unidentifiable. There are two important considerations in probing path selection, i.e., *minimizing probing cost* and *achieving identifiability*. Most prior works focus on minimizing the probing cost, but ignore identifiability. Consequently, the selected probing paths cannot uniquely infer the performance of the network links.

In this work, we address the problem of minimizing the probing cost and achieving identifiability in probe-based network link monitoring. Given a set of links to monitor, our objective is to select the minimum number of probing paths that can uniquely determine all identifiable links and cover all unidentifiable links. We propose an algorithm based on a linear system model to find out all irreducible sets of probing paths which can uniquely determine an identifiable link, and we develop an extended bipartite model to reflect the relationship between a set of probing paths and an identifiable link. Since our optimization problem is NP-hard, we propose a heuristic-based algorithm to greedily select probing paths. Our method eliminates two types of redundant probing paths, i.e., those that can be replaced by others and those without any contribution to achieving identifiability. Simulations based on real network topologies show that our approach can achieve identifiability with very low probing cost. Compared with prior work, our method is more general and has better performance.

1.3.2 Detecting and Localizing Large-Scale Router Failures

Detecting the occurrence of large-scale router failures and localizing the failed routers are critical to enhancing network reliability. Reply-based probe such as `ping` and `traceroute` are widely used for detecting and localizing failures in the Internet. However, existing works on failure detection only consider sporadic and isolated link failures, and hence are not suitable for large-scale router failures.

We propose a two-phase approach for detecting and localizing large-scale router failures using traceroute-like active probes. To detect large-scale router failures,

the detection phase is periodically invoked to probe all routers. When detecting large-scale router failures, the localization phase is triggered to identify the failed routers. We reduce the probing cost by avoiding three types of useless probes. For the routers whose status cannot be determined by probes, we develop a distance-based model to estimate their failure probability. Experimental results based on ISP topologies show that the accuracy of our approach is higher than 96.5%, even when only 10% of routers are connected by end systems for probing. Compared with prior works, the proposed approach achieves much higher accuracy with lower probing cost.

1.3.3 Optimal Recovery from Geographically Correlated Failures

Many events like natural disasters and intentional attacks can cause geographically correlated failures, i.e., routers and links within a geographically contiguous area all fail. The traditional backup path-based protection cannot effectively deal with geographically correlated failures, because links and their backup paths may fail simultaneously.

We propose an approach called Reactive Two-phase Rerouting (RTR) for intra-domain routing to quickly recover from geographically correlated failures with the shortest recovery paths. To recover a failed routing path, RTR first forwards packets around the failure area to collect information on failures. Then, in the second phase, RTR calculates a new shortest path and forwards packets along it through source routing. RTR can deal with geographically correlated failures associated with areas of any shape and location, and is free of permanent loops. For any failure area, the recovery paths provided by RTR are guaranteed to be the shortest. Extensive simulations based on ISP topologies show that RTR can find the shortest recovery paths for more than 98.6% of failed routing paths with reachable destinations. Compared with prior works, RTR achieves better performance for recoverable failed routing paths and uses much less network resources for irrecoverable failed routing paths.

1.3.4 Cross-Layer Approaches for Recovery from Correlated Link Failures

Backup paths are widely used to protect IP links from failures. Existing solutions such as the commonly used independent and Shared Risk Link Group models do not accurately reflect the correlation between IP link failures, and thus may not choose reliable backup paths. Moreover, most prior works consider backup path selection as a connectivity problem, but ignore the traffic load and bandwidth constraint of IP links.

We propose two cross-layer approaches to protect IP links with backup paths. In the first approach, we develop a correlated failure probability (CFP) model based on the topology mapping and failure probability of fiber links. The CFP model quantifies the impact of IP link failure on the reliability of backup paths. With the CFP model, we propose two algorithms for selecting a backup path to protect each IP link. The first algorithm focuses on choosing the backup paths with minimum failure probability. The second algorithm further considers the bandwidth constraint and aims at minimizing the traffic disruption caused by failures. Furthermore, it controls the rerouted traffic load not to exceed the usable bandwidth of IP links to avoid interfering with the normal traffic. In the second approach, we develop a probabilistically correlated failure (PCF) model based on the discovery of the recent Internet measurements. The PCF model considers both independent and correlated IP link failures, and quantifies the impact of IP link failure on the reliability of backup paths. With the PCF model, we propose an algorithm to choose multiple reliable backup paths to protect each IP link. When an IP link fails, its traffic is split onto multiple backup paths to ensure that the rerouted traffic load on each IP link does not exceed the usable bandwidth. We evaluate the proposed approaches using real ISP networks with both optical and IP layer topologies. Compared with existing works, the backup paths selected by our approaches are much more reliable. Moreover, the proposed approaches achieve higher recovery rate without causing link overload.

1.4 Organization

The remainder of the dissertation is organized as follows. Chapter 2 presents our approach for monitoring network link performance using end-to-end probe. Chapter 3 focuses on detecting the occurrence of large-scale router failures and localizing the failed routers using traceroute-like probes. Chapter 4 introduces an approach for intra-domain routing to quickly recover from geographically correlated failures with the shortest recovery paths. Chapter 5 presents two cross-layer approaches for IP link protection. Finally, we conclude the dissertation and discuss the future work in Chapter 6.

Chapter 2

Probe-Based Network Link Monitoring

2.1 Introduction

Continuously monitoring link performance is important for detecting link failures. Due to various advantages, tomography-based end-to-end probe has received much attention and has been widely used in network monitoring [19, 20, 21, 22, 23, 24, 25, 26, 27, 30, 31]. In this approach, end systems are connected to the network as shown in Fig. 2.1. An end system sends probing packets to another end system to measure the delay or loss rate of the routing path. Unlike Simple Network Management Protocol (SNMP)-based polling [44], end-to-end probe does not need to run agents on routers. In particular, it is suitable for monitoring the performance of network links that belong to a non-cooperative administrative domain, in which directly measuring link performance is usually hard to achieve [45]. Furthermore, compared with reply-based probe such as `ping` and `traceroute`, end-to-end probe uses normal data packets, and thus it does not have the problem of being ignored by intermediate routers [46] or blocked by firewalls [47].

Selecting probing paths is the major problem of probe-based network link monitoring. Generally, there are two important considerations, i.e., *minimizing probing cost* and *achieving identifiability*. The probing cost is mainly defined as the number of selected probing paths [21, 48, 49, 50, 51, 28]. It can also be the number of

end systems used for probing [27, 48, 49] and the cost specified by network components [21, 25]. Since probing traffic is periodically injected into the network, it consumes network bandwidth and increases workload of the routers. Lower probing cost means less resource consumption, less negative impact on the normal data transmission, and better scalability of the probing scheme. Therefore, many existing works [21, 25, 27, 48, 49, 50] focus on minimizing the probing cost in network link monitoring.

However, most prior works ignore identifiability and the selected probing paths cannot uniquely infer the performance of the network links. Since a probing path may consist of multiple links, a probe measures the performance of the whole probing path, and it cannot infer the performance of a specific link. Hence, selecting probing paths to cover a link may not be sufficient for monitoring the performance of the link. To uniquely infer the performance of a link, multiple coordinated probes are needed. Unfortunately, in a general network the performance of some links may not be able to be uniquely inferred from probes [52].

Network links can be classified into two types. If the performance of a link can be uniquely inferred by a set of probes, this link is *identifiable* and the set of probes can *uniquely determine* it. Otherwise, the link is *unidentifiable*. If probing paths are not properly selected, the performance of a link cannot be uniquely inferred, even if it is an identifiable link. As shown in Fig. 2.1, links e_1, e_2, and e_3 are identifiable, but links e_4 and e_5 are unidentifiable because every probe traversing e_4 also traverses e_5. Using probing paths p_1, p_2, and p_4 together can uniquely determine e_1, e_2, and e_3. However, only choosing p_1 and p_2 cannot uniquely determine any link.

In this chapter, we address the problem of minimizing the probing cost and achieving identifiability in probe-based network link monitoring. Given a set of links called *target links*[1] to monitor, our objective is to select the minimum number of probing paths that can uniquely determine all identifiable target links and cover all unidentifiable target links. There are three major challenges. First, a target link may be unidentifiable. Using multiple probing paths to monitor an unidentifiable link only wastes network bandwidth. Hence, it is necessary to differentiate an

[1]Target links can be links at critical topological location, and their performance usually affects a large area of the network [7].

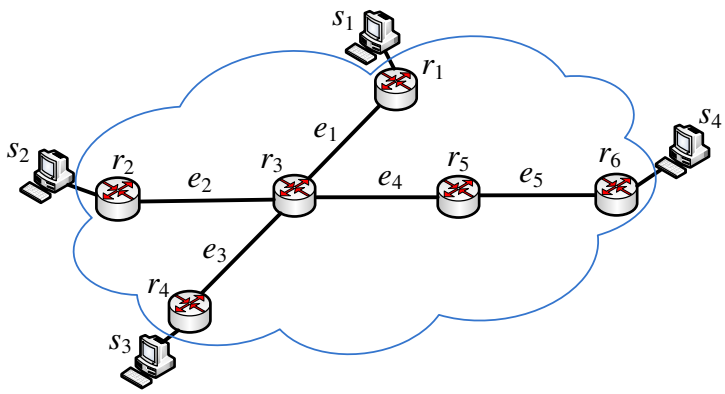


Figure 2.1. An example of network link monitoring with six probing paths $p_1(s_1, s_2)$, $p_2(s_1, s_3)$, $p_3(s_1, s_4)$, $p_4(s_2, s_3)$, $p_5(s_2, s_4)$, and $p_6(s_3, s_4)$, where $p_i(s_j, s_k)$ is the probing path between end systems s_j and s_k.

identifiable link from an unidentifiable link. Second, different probing paths have different contributions to achieving our objective, and we need to determine the most useful probing paths. Third, a probing path may be replaced by other probing paths [19, 50], and thus we should avoid selecting redundant probing paths so as to minimize the probing cost.

The basic idea of our approach is to only select probing paths that are the most useful for achieving our objective. More specifically, the proposed approach consists of three parts. First, we adopt a linear system to model the relationship between probing paths and links, and then find out identifiable target links by solving the linear system. Second, we design a method based on matrix decomposition and linear replacement to calculate irreducible sets of probing paths which can uniquely determine each identifiable target link. Third, we extend the bipartite model, which is commonly used for modeling the relationship between a single probing path and a link, to reflect the relationship between a set of probing paths and a link. With this model, we prove that our optimization problem is NP-hard, and then propose a heuristic-based algorithm to greedily select probing paths. Simulations based on real ISP networks show that our approach can achieve identifiability with very low probing cost. Compared with prior work, our method is more general and has better performance.

The rest of this chapter is organized as follows. We present the linear system model and the problem description in Section 2.2. Our approach is introduced

in Section 2.3, which includes the algorithm for calculating all irreducible sets of probing paths to uniquely determine an identifiable target link, the extended bipartite model, and the heuristic-based algorithm for selecting probing paths. Section 2.4 presents the performance evaluation, and Section 2.5 reviews related work. Finally, Section 2.6 concludes the chapter.

2.2 Preliminaries

In this section, we first introduce the system model and define our problem, and then describe the linear system model used in our approach.

2.2.1 System Model and Problem Description

Similar to prior works on network monitoring [21, 48, 53, 54, 55, 56, 57], we model the network as a connected undirected graph $G(V, E)$, where V is the set of nodes (routers) and $E = \{e_i | 1 \leq i \leq m\}$ is the set of edges (communication links between routers). In the rest of this chapter, we use edge and link interchangeably. Note that the proposed technique also works for asymmetric links, where the network is modeled as a directed graph.

Some routers in the network can be directly connected by end systems which can send and receive probing packets. The probing packet from end system s_j to end system s_k traverses the routing path from s_j to s_k. Such an end-to-end path is referred to as a *probing path*. A probing path can *cover* a link if it traverses this link. There are two types of links on each probing path. The first is the link between an end system and a router. In the Internet, end systems can only directly connect to edge routers and they are usually close to these routers, e.g., in the same building or campus. Thus, the links between end systems and edge routers are quite short. The second is the link between two routers, which is usually hundreds of miles long in the current Internet. Since the performance of the first type of links is usually quite stable, they are commonly omitted in network link monitoring. Hence, we only count the performance of the second type of links. For example, in Fig. 2.1 the probe from s_1 to s_3 measures the delay of the probing path p_2. We omit the delay of the link from s_1 to r_1 and the link from r_4 to s_3.

Consequently, the measured delay of probing path p_2 is caused by links e_1 and e_3.

In the network, there are n probing paths $P = \{p_i | 1 \leq i \leq n\}$ which can be used for monitoring a set of m_t target links $E_t \subseteq E$. For simplicity, target links are labeled as $e_1, \cdots, e_{m_t}$. We assume that the network topology is available, which is a widely used assumption in probe-based network monitoring. Based on the network topology, we know which links can be covered by a probing path. Network monitoring is very important for Internet Service Providers (ISPs), because they need to keep track of the performance of their networks. ISPs know the complete topology of their networks, and thus this approach has been used in practice.

Probe-based network link monitoring has two steps. The first is to select a set of probing paths. Then, end systems on the chosen probing paths periodically issue probing packets and send measurement results to the central Network Operations Center (NOC). We focus on the first step; i.e., selecting probing paths. We define the problem of minimizing the probing cost and achieving identifiability as follows, where the probing cost is defined as the number of probing paths selected for network monitoring.

Definition 1 (Problem definition)**.** *Given a network G, a set of probing paths P, and a set of target links E_t, the objective is to select the minimum number of probing paths from P, such that all identifiable target links can be uniquely determined and all unidentifiable target links are covered.*

Minimizing probing cost is an important objective in existing approaches of probe-based network monitoring. Lower probing cost means less resource consumption, less negative impact on the normal data transmission, and better scalability. In most prior works [21, 48, 49, 50, 51, 28], the probing cost is defined as the number of selected probing paths. Therefore, we aim at choosing a minimum set of probing paths for link monitoring.

The goal of link monitoring is to detect and localize the target links with abnormal performance. Hence, it tries to infer the performance of each target link from probing results as accurately as possible. For identifiable target links, the performance can be uniquely inferred from probes. Hence, we intend to select probing paths to uniquely determine them. If any of them has abnormal performance, we can accurately identify the link. For unidentifiable target links, the performance

cannot be uniquely inferred from probes. We can estimate their performance with techniques like second-order statistics [26, 30]. Although the estimated performance may not be accurate, it is still useful for link performance diagnosis. To use the second-order statistics technique, each unidentifiable target link should be covered by at least one probing path. Hence, we also select probing paths to cover all unidentifiable target links.

2.2.2 Linear System Model

For link performance satisfying the additive metric, the relationship between probing paths and links can be naturally modeled as a linear system $LS = \{ls_i | 1 \leq i \leq n\}$, in which ls_i is the ith linear equation as shown in Eq. (2.1). Binary variable a_{ij} is 1 if probing path p_i covers link e_j; otherwise it is 0. Variables x_j and b_i represent the performance of link e_j and probing path p_i. Thus, Eq. (2.1) means that the performance of p_i is the addition of the performance of all links on p_i.

$$\sum_{j=0}^{m} a_{ij} x_j = b_i \tag{2.1}$$

The linear system can be written in the matrix computation form as shown in Eq. (2.2). Variable $\mathbf{x}$ is the vector form of $x_1, \cdots, x_m$ and $\mathbf{b}$ is the vector form of $b_1, \cdots, b_n$. The coefficient matrix $A = (\mathbf{a}_1, \cdots, \mathbf{a}_n)^T$ is also called the *dependency matrix* [49].

$$A\mathbf{x} = \mathbf{b} \tag{2.2}$$

The linear system model is suitable for delay and loss rate which are major link performance metrics in network link monitoring. We use delay as an example to explain the meaning of the linear equations. The network in Fig. 2.1 has six probing paths and five links. Hence, the linear system has six equations and five variables $x_1, \cdots, x_5$ as shown in Fig. 2.2(a). The dependency matrix is shown in Fig. 2.2(b). For probing path p_1 between end systems s_1 and s_2, the overall delay of this path is the sum of the delay on links e_1 and e_2. Therefore, in the first

$$\begin{cases} x_1 + x_2 = b_1 \\ x_1 + x_3 = b_2 \\ x_1 + x_4 + x_5 = b_3 \\ x_2 + x_3 = b_4 \\ x_2 + x_4 + x_5 = b_5 \\ x_3 + x_4 + x_5 = b_6 \end{cases} \qquad A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

(a) The linear system model LS (b) The dependency matrix

Figure 2.2. The linear system model and the dependency matrix of the example in Fig. 2.1.

equation in Fig. 2.2(a), b_1 is equal to the sum of x_1 and x_2.

For loss rate, we need to make a transformation with logarithm. Suppose r_j is the loss rate of link e_j and q_i is the loss rate of link e_j. The loss rate satisfies $\prod_{j=0}^{m}(1 - a_{ij}r_j) = 1 - q_i$. By applying logarithm to both sides, we have $\lg \prod_{j=0}^{m}(1-a_{ij}r_j) = \lg(1-q_i)$, which results in $\sum_{j=0}^{m} \lg(1-a_{ij}r_j) = \lg(1-q_i)$. There are two cases for the value of $1 - a_{ij}r_j$. If $a_{ij} = 1$, i.e., probing path p_i covers link e_j, $1 - a_{ij}r_j = 1 - r_j$. If $a_{ij} = 0$, we have $1 - a_{ij}r_j = 1$, and thus $\lg(1 - a_{ij}r_j) = 0$. For each probing path p_i, we define $b_i = \lg(1 - q_i)$. Similarly, for each link e_j, we let $x_j = \lg(1 - r_j)$ if e_j is on p_i, or $x_j = 0$ otherwise. Through this transformation, we can apply the linear equation in Eq. (2.1) to loss rate. For example, the loss rate of p_1 in Fig. 2.1 is $q_1 = 1-(1-r_1)(1-r_2)$. With the transformation, we have $\lg(1 - r_1) + \lg(1 - r_2) = \lg(1 - q_1)$. By defining $x_1 = \lg(1 - r_1)$, $x_2 = \lg(1 - r_2)$, and $b_1 = \lg(1 - q_1)$, this equation becomes $x_1 + x_2 = b_1$ which is the first equation in Fig. 2.2(a).

The above transformation is not suitable for the probing paths that traverse failed links. Suppose the probing path p_i that traverses link e_j and e_j fails. The loss rates of p_i and e_j will both be 1, and thus $\lg(1 - q_i)$ and $\lg(1 - r_j)$ are not defined. To solve this problem, we simply ignore the probing result of p_i, if the measured loss rate of p_i is 1. It equals to removing a linear equation from the linear system. In practice, probing paths are very unlikely to traverse failed links due to two reasons. First, the failure probability of links in the Internet is very low and most link failures are transient (about 50% of them last for less than 1 minute and 90% of them are shorter than 10 minutes [17]). Hence, probing paths are free

of link failures most of time. Second, when long-lasting link failures occur, the routing protocol can quickly converge to a new network topology, and all routing paths can bypass the failed links. In this case, we need to select probing paths based on the new network topology.

We can use the linear system model to determine if a link is identifiable and if a probing path can be replaced by other probing paths. First, a link e_i is identifiable if and only if the corresponding variable x_i in the linear system is solvable. Hence, we can determine all identifiable links by solving the linear system. Second, a probing path p_i can be replaced by a set of probing paths, if row vector $\mathbf{a}_i$ in the dependency matrix can be linearly expressed by the row vectors corresponding to this set of probing paths. In Fig. 2.2(b), row vector $\mathbf{a}_5$ can be linearly expressed by row vectors $\mathbf{a}_2$, $\mathbf{a}_3$, and $\mathbf{a}_4$ as $\mathbf{a}_5 = -\mathbf{a}_2 + \mathbf{a}_3 + \mathbf{a}_4$. It means that probing paths p_2, p_3, and p_4 together can replace probing path p_5.

Due to linear dependence among row vectors, there may be many sets of linear equations for a solvable variable. Accordingly, an identifiable link may be uniquely determined by multiple sets of probing paths. We define the solution to an identifiable link as follows.

Definition 2 (Solution to an identifiable link)**.** *A* solution *to an identifiable link e_i is an irreducible set of probing paths that can uniquely determine e_i.*

Irreducible means that each probing path in the set cannot be replaced by other probing paths in the same set. This definition is consistent with our objective of minimizing the number of selected probing paths. Consider a set of probing paths that can uniquely determine link e_i. If a probing path can be replaced by others in the same set, the set can still uniquely determine e_i after removing this probing path. For uniquely determining e_i, containing such replaceable probing paths only increases the probing cost. Therefore, we require that a solution does not contain redundant probing paths. For example, x_1 in Fig. 2.2(a) is a solvable variable and $\frac{1}{2}b_1 + \frac{1}{2}b_2 - \frac{1}{2}b_4$ is a solution to x_1. It contains three variables b_1, b_2, and b_4, and thus probing paths p_1, p_2, and p_4 together can uniquely determine link e_1. Since row vectors $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_4$ are linearly independent, each probing path is not replaceable by other two probing paths. Therefore, set $\{p_1, p_2, p_4\}$ is a solution to e_1.

Note that a solution to a solvable variable x_i in the linear system does not necessarily match a solution to an identifiable link e_i. A solution to x_i is a linear combination of variable b_js. The corresponding probing paths may contain replaceable ones. For example, the linear expression $b_1 - \frac{1}{2}b_2 + \frac{1}{2}b_3 - b_5 + \frac{1}{2}b_6$ is a solution to x_1 in Fig. 2.2(a). However, the set of probing paths $\{p_1, p_2, p_3, p_5, p_6\}$ is not a solution to link e_1, because the corresponding five row vectors are not linearly independent. Table 2.1 summarizes the symbols used in the linear system model and the symbols that will be used in the path selection algorithm in the next section.

Table 2.1. Table of notations

Symbols	**Meaning**
G	network under study
m	the number of links in G
e_i	the ith link in G
n	the number of probing paths in G
p_i	the ith probing path in G
E_t	set of target links
m_t	the number of target links
LS	linear system model of G
ls_i	the ith linear equation of LS
b_i	measured performance of the ith probing path
A	dependency matrix
S^i	set of all solutions to the solvable variable x_i in LS
S^i_j	the jth solution to the solvable variable x_i in LS
L	set of linear expressions, $L = \{l_i \mid 1 \leq i \leq t\}$
l_i	linear expression for the ith row vector in A_N

2.3 Path Selection Algorithm

This section introduces our algorithm for selecting probing paths.

2.3.1 Overview

The basic idea of our algorithm is to determine the contribution of a probing path for achieving our objective, and then choose the most useful probing paths

and avoid selecting redundant ones. Probing paths have different contributions to uniquely determine identifiable links and cover unidentifiable links. An identifiable link may have many solutions, and a probing path may be in the solutions to multiple identifiable links. Similarly, an unidentifiable link may be covered by several probing paths, and a probing path can cover multiple links.

The linear system model proposed in Section 2.2.2 can reflect the contribution of a probing path to covering unidentifiable links. For identifiable links, a natural method is to find out all solutions to each identifiable link, and thus we can know which probing paths are the most useful to achieving identifiability. We propose a method based on matrix decomposition and linear replacement to calculate all solutions to an identifiable link in Section 2.3.2 and Section 2.3.3. Moreover, we develop an extended bipartite model to reflect the relationship between probing paths and target links in Section 2.3.4. Through this model, we introduce a heuristic-based algorithm in Section 2.3.5 which can efficiently select probing paths to achieve our objective.

2.3.2 Decomposition of Linear System

The first step of our approach is to determine if a link is identifiable or not. Solving the linear system is the simplest way to achieve this. Many techniques in linear algebra, such as Gaussian elimination, can solve the linear system. A key observation is that the linear dependence between row vectors of the dependency matrix only affects how many solutions an identifiable link has, but does not affect if a link is identifiable or not. Therefore, we decompose the dependency matrix, based on which we can determine all identifiable target links and calculate one solution for each of them. Then, we use linear replacement to calculate all solutions, which will be introduced in the next subsection.

To decompose the dependency matrix, we divide its row vectors into two groups as shown in Eq. (2.3). A_R is a maximal independent set of row vectors of matrix A. Suppose A_R contains r row vectors. Matrix A_N contains the other $n-r$ row vectors. According to linear algebra theory, each row vector of A_N can be linearly expressed by row vectors of A_R. For simplicity, we renumber the probing paths such that $A_R = (\mathbf{a}_1, \cdots, \mathbf{a}_r)^T$ and $A_N = (\mathbf{a}_{r+1}, \cdots, \mathbf{a}_n)^T$.

$$A = \begin{pmatrix} A_R \\ A_N \end{pmatrix} \tag{2.3}$$

We use the algorithm introduced in [50] to decompose the dependency matrix, which is based on standard rank-revealing decomposition techniques [58]. A dependency matrix may have multiple decompositions. Our algorithm for calculating all solutions for each identifiable target link does not have any requirement on the matrix decomposition. Therefore, we can use any tie breaking strategy to choose a decomposition. Based on the matrix decomposition, the linear system in Eq. (2.2) is also decomposed into two parts as shown in Eq. (2.4), in which vector $\mathbf{b}$ is partitioned into two vectors $\mathbf{b}_R = (b_1, \cdots, b_r)^T$ and $\mathbf{b}_N = (b_{r+1}, \cdots, b_n)^T$.

$$\begin{pmatrix} A_R \\ A_N \end{pmatrix} \mathbf{x} = \begin{pmatrix} \mathbf{b}_R \\ \mathbf{b}_N \end{pmatrix} \tag{2.4}$$

Since each row vector of A_N is a linear combination of row vectors of A_R, a solvable/unsovlable variable in linear system $A_R\mathbf{x} = \mathbf{b}_R$ is also solvable/unsolvable in linear system $A\mathbf{x} = \mathbf{b}$. By solving linear system $A_R\mathbf{x} = \mathbf{b}_R$, we can determine whether a target link is identifiable or not. Also, we calculate a solution S_1^i to each solvable variable x_i, which is referred to as the *base solution*. Solving linear system $A_R\mathbf{x} = \mathbf{b}_R$ can obtain only one base solution to each solvable variable x_i, because the row vectors of A_R are linearly independent.

Next, we compute the relationship between variable b_is. Each row vector of A_N can be linearly expressed by row vectors of A_R. By solving the linear system in Eq. (2.5), we can calculate the linear expression for row vector $\mathbf{a}_{r+i}$ of A_N, where $i = 1, \cdots, n-r$.

$$\sum_{j=1}^{r} c_{ij}\mathbf{a}_j + \mathbf{a}_{r+i} = 0 \tag{2.5}$$

Multiplying vector $\mathbf{x}$ to both sides of Eq. (2.5) results in $\sum_{j=1}^{r} c_{ij}\mathbf{a}_j\mathbf{x} + \mathbf{a}_{r+i}\mathbf{x} =$

0. Since $\mathbf{a}_j\mathbf{x} = b_j$ for $1 \leq j \leq r$ and $\mathbf{a}_{r+i}\mathbf{x} = b_{r+i}$, we have the linear expression in Eq. (2.6). Through it, we obtain the relationship between b_is, which will be used for calculating all solutions to identifiable target links. We use $L = \{l_i | 1 \leq i \leq n-r\}$ to represent the set of these linear expressions, where l_i is the expression containing b_{r+i}.

$$\sum_{j=1}^{r} c_{ij} b_j + b_{r+i} = 0 \tag{2.6}$$

The dependency matrix in Fig. 2.2(b) shows an example. A decomposition of this matrix is $A_R = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4)^T$ and $A_N = (\mathbf{a}_5, \mathbf{a}_6)^T$. The corresponding partition of vector $\mathbf{b}$ is $\mathbf{b}_R = (b_1, b_2, b_3, b_4)^T$ and $\mathbf{b}_N = (b_5, b_6)^T$. By solving linear system $A_R\mathbf{x} = \mathbf{b}_R$, we discover that links e_1, e_2, and e_3 are identifiable but links e_4 and e_5 are unidentifiable. The basic solution to x_1 is $\frac{1}{2}b_1 + \frac{1}{2}b_2 - \frac{1}{2}b_4$, which means that set $\{p_1, p_2, p_4\}$ is a solution to link e_1. Moreover, set L contains two linear expressions, i.e., l_1 is $b_2 - b_3 - b_4 + b_5 = 0$ and l_2 is $b_1 - b_3 - b_4 + b_6 = 0$.

2.3.3 Solution Calculation

In this subsection, we propose an algorithm to calculate all solutions to a solvable variable, through which we can obtain all solutions to each identifiable target link. The basic idea of calculating all solutions to a solvable variable x_i is to replace the variables in solutions with the linear expressions in set L. Initially, we have only the base solution S_1^i obtained from linear system $A_R\mathbf{x} = \mathbf{b}_R$. For each linear expression $l_j \in L$, if it has a common variable b_k with S_1^i, replacing b_k in S_1^i with l_j results in a linear expression that can solve x_i. The variables in this linear expression correspond to a set of probing paths which can uniquely determine link e_i. We check if the set contains any replaceable probing paths. If not, this set of probing paths is a solution to e_i. Similar linear replacement is applied to the obtained solutions, until no new solution can be found.

The algorithm for calculating all solutions to a solvable variable x_i is shown in Algorithm 1. The input is base solution S_1^i and set L of linear expressions. We use a queue to store the solutions. As shown in line 5–14, each time we take out

a solution S_c^i from the queue and apply all possible linear replacements to it. A linear replacement generates a linear expression S_t^i (line 8). Then, in line 9, the algorithm checks if each variable in S_t^i is replaceable by other variables in S_t^i. If not, S_t^i corresponds to a solution to link e_i. We record S_t^i by putting it into queue Q, if it is not in queue Q and set S^i. After applying all linear replacements to the current solution S_c^i, we put it into set S^i. As a result, at any moment set S^i contains all solutions that have been applied linear replacements, and queue Q contains all solutions that will be applied linear replacements. When queue Q becomes empty, the algorithm stops and all solutions to x_i are in set S^i.

Algorithm 1 SolutionCalculation

Input: Base solution S_1^i, and set L of linear expressions
Output: Set S^i containing all solutions to x_i
Procedure:

```
1: INIT(Q)    ▷ Initialize a queue Q
2: S^i ← ∅
3: ENQUEUE(Q, S_1^i)
4: while Q ≠ ∅ do
5:    S_c^i ← DEQUEUE(Q)
6:    for each linear expression l_j ∈ L do
7:       for each common variable b_k in S_c^i and l_j do
8:          S_t^i ← replace b_k in S_c^i with l_j
9:          if each variable in S_t^i cannot be replaced by other variables in S_t^i, and
            S_t^i ∉ Q ∧ S_t^i ∉ S^i then
10:            ENQUEUE(Q, S_t^i)
11:         end if
12:      end for
13:   end for
14:   S^i ← S^i ∪ S_c^i
15: end while
```

Theorem 1. *The algorithm SolutionCalculation can find all solutions to a solvable variable in the linear system. (The proof is given in Appendix A.)*

A solution to variable x_i is a linear combinations of $b_1, \cdots, b_n$. Since b_j corresponds to probing path p_j, we can easily obtain the corresponding solution to identifiable link e_i. Based on Theorem 1, our algorithm can find all solutions to each identifiable link. The computational complexity of this algorithm is determined by the number of linear replacements, which may not be a linear function

of the size of dependency matrix. For a large-scale network, an identifiable link may have a large number of solutions, and thus the algorithm needs to run quite long. Actually, the algorithm for selecting probing paths does not need to use all solutions. For an identifiable link, we can choose some of its solutions for probing path selection, and hence the algorithm only needs to calculate some solutions rather than all solutions. More details will be introduced in Section 2.3.5.

To make it more clear, we use an example to show how to calculate all solutions to identifiable link e_1 in Fig. 2.1. Initially, we have base solution $S_1^1 : \frac{1}{2}b_1+\frac{1}{2}b_2-\frac{1}{2}b_4$. Set L contains two linear equations $l_1 : b_2-b_3-b_4+b_5 = 0$ and $l_2 : b_1-b_3-b_4+b_6 = 0$. The base solution has two common variables b_2 and b_4 with linear expression l_1. Replacing b_2 in S_1^1 with $b_3 + b_4 - b_5$ results in $\frac{1}{2}b_1 + \frac{1}{2}b_3 - \frac{1}{2}b_5$, which is a new solution to x_1. We name it as S_2^1. Similarly, replacing b_4 in S_1^1 with $b_2 - b_3 + b_5$ produces $\frac{1}{2}b_1 + \frac{1}{2}b_3 - \frac{1}{2}b_5$. Then we use l_2 to replace b_1 and b_4 in S_1^1. Both of them result in $\frac{1}{2}b_2 + \frac{1}{2}b_3 - \frac{1}{2}b_6$, which is named as S_3^1. Until now, we have already applied all possible linear replacements to S_1^1. Next, we apply linear replacements to S_2^1 and S_3^1. The generated new solutions are put into the queue, and linear replacements continue, until the queue is empty. When applying l_1 to solution $b_1 - \frac{1}{2}b_4 - \frac{1}{2}b_5 + \frac{1}{2}b_6$ to replace b_4, we get linear expression $b_1 - \frac{1}{2}b_2 + \frac{1}{2}b_3 - b_5 + \frac{1}{2}b_6$. Since the corresponding row vectors $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_5, \mathbf{a}_6\}$ are not linearly independent, we do not take it as a solution to x_i. The algorithm stops after finding out six solutions listed in the first column of Table 2.2. Accordingly, identifiable link e_1 has six solutions listed in the second column of Table 2.2.

Table 2.2. Solutions to variable x_1 in linear system and corresponding solutions to link e_1

Solution to x_1	**Solution to e_1**
$\frac{1}{2}b_1 + \frac{1}{2}b_2 - \frac{1}{2}b_4$	$\{p_1, p_2, p_4\}$
$\frac{1}{2}b_1 + \frac{1}{2}b_3 - \frac{1}{2}b_5$	$\{p_1, p_3, p_5\}$
$\frac{1}{2}b_2 + \frac{1}{2}b_3 - \frac{1}{2}b_6$	$\{p_2, p_3, p_6\}$
$b_3 + \frac{1}{2}b_4 - \frac{1}{2}b_5 - \frac{1}{2}b_6$	$\{p_3, p_4, p_5, p_6\}$
$b_1 - \frac{1}{2}b_4 - \frac{1}{2}b_5 + \frac{1}{2}b_6$	$\{p_1, p_4, p_5, p_6\}$
$b_2 - \frac{1}{2}b_4 + \frac{1}{2}b_5 - \frac{1}{2}b_6$	$\{p_2, p_4, p_5, p_6\}$

2.3.4 Extended Bipartite Model

As in the literature [24], the relationship between probing paths and target links is usually modeled as a bipartite graph $G_B = (U, V, E)$, where U and V are two sets of vertices and E is a set of edges. Vertex $u_i \in U$ is probing path p_i, and $v_j \in V$ is target link e_j. If probing path p_i traverses target link e_j, the bipartite graph has an edge between u_i and v_j.

The traditional bipartite model reflects the coverage relation between a probing path and a target link. However, it cannot reflect which probing paths together can uniquely determine an identifiable target link. We propose an extended bipartite model to address this problem. Our extended bipartite model $G'_B = (U, V, E)$ has vertex sets U and V and edge set E. Vertex set U has two subsets U_1 and U_2, and vertex set V also has two subsets V_1 and V_2 as shown in Fig. 2.3. A vertex in set V_1 represents an identifiable target link, and a vertex in V_2 represents an unidentifiable target link. Each vertex in U_1 represents a solution to an identifiable target link, and hence it corresponds to a set of probing paths. Also, we use a vertex in U_2 to represent a probing path that can cover unidentifiable target links. Therefore, set E contains three kinds of edges as follows.

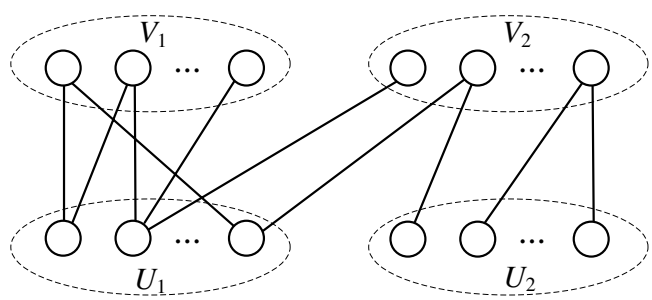


Figure 2.3. An illustration of the extended bipartite model.

1. The edge between V_1 and U_1. For an identifiable target link represented by vertex v_i^1 in V_1, if vertex u_j^1 in U_1 represents a solution to this link, there is an edge connecting v_i^1 and u_j^1. The edge reflects the identifiability relationship between a solution and an identifiable target link.

2. The edge between V_2 and U_1. For an unidentifiable target link represented by vertex v_i^2 in V_2, if the probing paths represented by vertex u_j^1 in U_1 can

cover this link, there is an edge between v_i^2 and u_j^1. The edge reflects the coverage relation between a solution and an unidentifiable target link.

3. The edge between V_2 and U_2. For an unidentifiable target link represented by vertex v_i^2 in V_2, if the probing path represented by vertex u_j^2 in U_2 can cover this link, there is an edge between v_i^2 and u_j^2. The edge reflects the coverage relation between a single probing path and an unidentifiable target link.

There is no edge between sets V_1 and U_2. For identifiable target links, we need to select probing paths to uniquely determine them, which is achieved by choosing their solutions from set U_1. Edges between V_1 and U_1 are sufficient for this task. Thus, we do not add any edge between an identifiable target link and a single probing path.

The commonly used bipartite model is a special case of our extended bipartite model. If we do not consider the identifiability, we can intentionally mark all target links as unidentifiable to make set V_1 empty. Accordingly, set U_1 becomes empty because there is no solution. Hence, the extended bipartite model turns into the commonly used bipartite model.

2.3.5 Path Selection Algorithm

Through the extended bipartite model, our path selection problem equals to selecting a set of vertices from $U_1 \cup U_2$, so that all vertices in $V_1 \cup V_2$ are their neighbors. We have the following theorem.

Theorem 2. *The path selection problem is NP-hard.*

Proof. The path selection problem can be proved to be NP-hard via proving that the set cover problem is a special case of this problem. In the extended bipartite model, vertex $u_i^1 \in U_1$ represents a set of probing paths and vertex $u_j^2 \in U_2$ represents a single probing paths. Consider the number of probing paths represented by a vertex in set U_1 and U_2, we have $|u_i^1| \geq 1$ and $|u_j^2| = 1$. By setting $|u_i^1|$ to 1 for each vertex $u_i^1 \in U_1$, it is equivalent to minimizing the number of vertices selected from $U_1 \cup U_2$, which is the same as the set cover problem. Therefore, the

set cover problem is a special case of our path selection problem. Since the set cover problem is NP-hard, our problem is also NP-hard. □

Since this problem is NP-hard, we propose a heuristic-based algorithm to solve it based on the extended bipartite model. The basic idea is to greedily select probing paths, until all identifiable target links can be uniquely determined and all unidentifiable target links are covered. The idea of greedy selection is widely used for probing path selection, because it has low computational complexity and usually achieves good performance. A large-scale network can have thousands of probing paths, and thus an identifiable target link may have a large number of solutions. To enhance the scalability of our approach, we introduce a parameter α to control how many solutions are used for an identifiable target link. The computational complexity of the algorithm directly relates to α. A smaller α makes the algorithm faster. On the other hand, a larger α is helpful for reducing the probing cost as shown in our evaluation in Section 2.4.3.

The path selection algorithm is shown in Algorithm 2. It takes linear system LS and parameter α as input, and returns a set of probing paths that can uniquely determine all identifiable target links and cover all unidentifiable target links. The algorithm first calculates solutions to each identifiable target link with the algorithm SolutionCalculation, and then selects α solutions for each of them. Next, it builds the extended bipartite graph. Then, the algorithm starts to select probing paths in the following three steps.

First, the algorithm deals with identifiable target links (line 8–16), because uniquely determining an identifiable link is more complex and needs more probing paths than covering an unidentifiable link. Consider an identifiable target link represented by vertex $v_i^1 \in V_1$. We choose one of its solutions to uniquely determine this link. In the extended bipartite graph, it is equivalent to choosing a vertex u_j^1 from U_1 such that v_i^1 and u_j^1 are neighbors. There are three considerations when selecting such a vertex v_i^1 from U_1: 1) v_i^1 may have multiple neighbors in V_1, i.e., this set of probing paths can uniquely determine multiple identifiable target links; 2) v_i^1 represents a set of probing paths, and our objective is to minimize the number of selected probing paths; 3) v_i^1 may have neighbors in set V_2, i.e., this set of probing paths cover some unidentifiable target links. We intend to add as few probing paths as possible to uniquely determine as many identifiable target

links as possible. Hence, the algorithm selects a vertex from U_1 with the maximal value of $\frac{numNewVertices1}{numNewPaths}$ in each round as shown in line 9. If there are multiple candidates in U_1, we further consider how many unidentifiable target links can be covered. This process continues until all identifiable target links can be uniquely determined.

Next, the algorithm selects probing paths for unidentifiable target links (line 17–25). When all identifiable target links are handled, unselected probing paths are contained in sets U_1 and U_2. Hence, the algorithm selects vertices from $U_1 \cup U_2$, until all vertices in V_2 have a neighbor in U_1 or U_2. We intend to add as few probing paths as possible to cover as many unidentifiable target links as possible. Accordingly, the algorithm chooses a vertex with the maximal value of $\frac{numNewVertices2}{numNewPaths}$ in each round as shown in line 18.

Finally, the algorithm removes all replaceable ones from the selected probing paths (line 26). In the first two steps, the probing paths selected in each round do not contain replaceable ones. However, this cannot ensure that the final set of selected probing paths does not contain replaceable probing paths. Therefore, after finishing probing path selection, the algorithm removes all replaceable ones from the selected probing paths. This is achieved by the linear system-based method introduced in Section 2.2.2.

Theorem 3 (Correctness)**.** *The set of probing paths returned by the algorithm PathSelection can uniquely determine all identifiable target links and cover all unidentifiable target links. (The proof is given in Appendix B.)*

Next, we present the performance bound of the path selection algorithm. An example in Section 2.4.5 will show that the number of selected probing path can be smaller than the row rank of the dependency matrix.

Theorem 4 (Performance bound)**.** *The upper bound of the number of probing paths selected by the algorithm PathSelection is the row rank of the dependency matrix.*

Proof. Consider set P_s before removing its replaceable probing paths, we have $P_s \subseteq P$. Similar to P, set P_s corresponds to a linear system LS_s in the form of Eq. (2.2). The dependency matrix A_s of LS_s is formed by some row vectors of

Algorithm 2 PathSelection

Input: The linear system LS and parameter α
Output: A set of probing paths P_s
Procedure:

for each identifiable target link **do**
  Calculate solutions to it with the algorithm SolutionCalculation, and then select α solutions.
end for
Construct the extended bipartite graph $G'_B = (U, V, E)$ with the selected solutions.
$P_s = \emptyset$
Mark all vertices in V_1, V_2, U_1, and U_2 as uncovered.
Mark all probing paths as unused
while V_1 has uncovered vertices **do**
  Select an uncovered vertex u_m^1 from U_1 with the largest $\frac{numNewVertices1}{numNewPaths}$, where $numNewVertices1$ is the number of uncovered vertex in V_1 connected to u_m^1, and $numNewPaths$ is the number of unused probing path contained in u_m^1. If there are multiple candidates, select the one that connects to the maximal uncovered vertices in V_2.
  Mark u_m^1 as covered.
  Mark all uncovered vertices in $V_1 \cup V_2$ connected to u_m^1 as covered.
  for each unused probing path p_i in u_m^1 **do**
    $P_s = P_s \cup p_i$
    Mark p_i as used.
  end for
end while
while V_2 has uncovered vertices **do**
  Select an uncovered vertex u_m^2 from $U_1 \cup U_2$ that has the largest $\frac{numNewVertices2}{numNewPaths}$, where $numNewVertices2$ is the number of uncovered vertex in V_2 connected to u_m^2, and $numNewPaths$ is the number of unused probing path contained in u_m^1.
  Mark u_m^2 as covered.
  Mark all uncovered vertices in V_2 connected to u_m^2 as covered.
  for each unused probing path p_j in u_m^2 **do**
    $P_s = P_s \cup p_j$
    Mark p_j as used.
  end for
end while
Remove all replaceable probing paths from set P_s.

the original dependency matrix A. Therefore, the row rank of A_s is no larger than that of A.

After removing all replaceable probing paths, the probing paths left in P_s correspond to another linear system LS'_s. Its dependency matrix A'_s is formed by the maximal independent set of row vectors of A_s. Therefore, the rank of A'_s is the same as that of A_s, which is no larger than the rank of matrix A.

In conclusion, the number of probing paths returned by the algorithm is no larger than the row rank of the dependency matrix. □

Finally, we use an example to show our algorithm selects probing paths for target links e_1 and e_4 in Fig. 2.1. In the example, we use all solutions for path selection. The corresponding bipartite graph is shown in Fig. 2.4. Set U_1 has six vertices, each of which represents a solution to identifiable target link e_1. Set U_2 has three vertices, because there are three probing paths p_3, p_5, and p_6 that traverse unidentifiable target link e_4. Vertex e_4 in V_2 has no edge to vertex $\{p_1, p_2, p_4\}$ in U_1, because these three probing paths do not traverse e_4. The algorithm first chooses a set of probing paths for e_1. Among all six vertices in U_1, vertices $\{p_1, p_2, p_4\}$, $\{p_1, p_3, p_5\}$, and $\{p_2, p_3, p_6\}$ have the same value of $\frac{numNewVertices1}{numNewPaths}$. Since $\{p_1, p_3, p_5\}$ and $\{p_2, p_3, p_6\}$ can also cover one unidentifiable link, the algorithm selects one of them, e.g., $\{p_1, p_3, p_5\}$. After choosing it, all vertices in $V_1 \cup V_2$ are marked as covered. Since the selected set does not contain any replaceable probing path, the algorithm stops and returns a set of probing paths $\{p_1, p_3, p_5\}$.

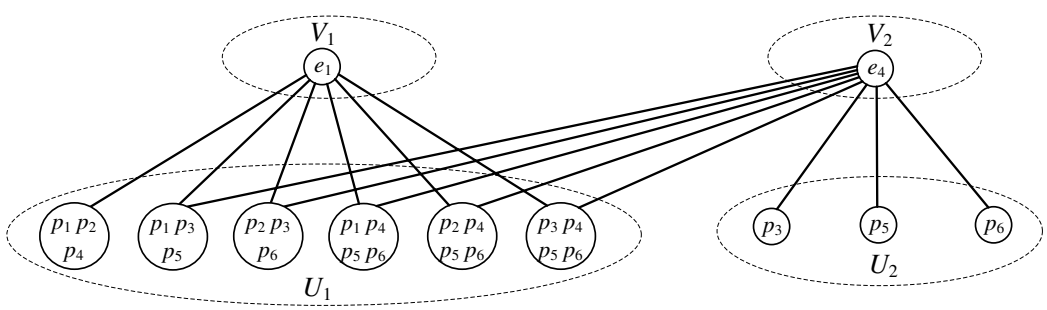


Figure 2.4. The extended bipartite graph for selecting probing paths for target links e_1 and e_4 in Fig. 2.1.

2.4 Performance Evaluation

This section evaluates the performance of our algorithm. We study how the performance is affected by network topologies, the percentage of nodes for probing, and the percentage of target links. Moreover, we compare the performance of our method with prior work [50].

2.4.1 Simulation Setup

The simulation is based on nine ISP topologies derived by the Rocketfuel project [59], which are widely used in evaluation of related works. Table 2.3 summarizes the number of nodes and links in each topology. All topologies adopt the shortest path routing calculated based on link cost. Since probes traverse routing paths, some links cannot be covered by probes. Hence, we select target links only from the links that can be covered by probing paths.

Table 2.3. Summary of topologies used in simulation

Topology	# Nodes	# Links
AS209	58	108
AS701	83	219
AS2914	70	111
AS3320	70	355
AS3356	63	285
AS3549	61	486
AS3561	92	329
AS4323	51	161
AS7018	115	148

We consider three parameters in the evaluation. The first is the percentage of target links. We randomly select 10% to 100% of the links that can be covered by probing paths as target links. The second is the percentage of *probers*, i.e., routers that are directly connected by end systems. We randomly select 20%, 40%, and 60% of routers as probers. The third is parameter α in the algorithm PathSelection. We set it to 1, 10, 100, and 1000. The path selection algorithm does not specify how to select α solutions; thus we randomly select α solutions in the evaluation. Different strategies for selecting α to build the extended bipartite

graph are left for future work. For each simulation, we run it 100 times and report the average result.

We define the following two metrics to quantify the performance.

1. *Overall probing cost*: It is the number of the selected probing paths.

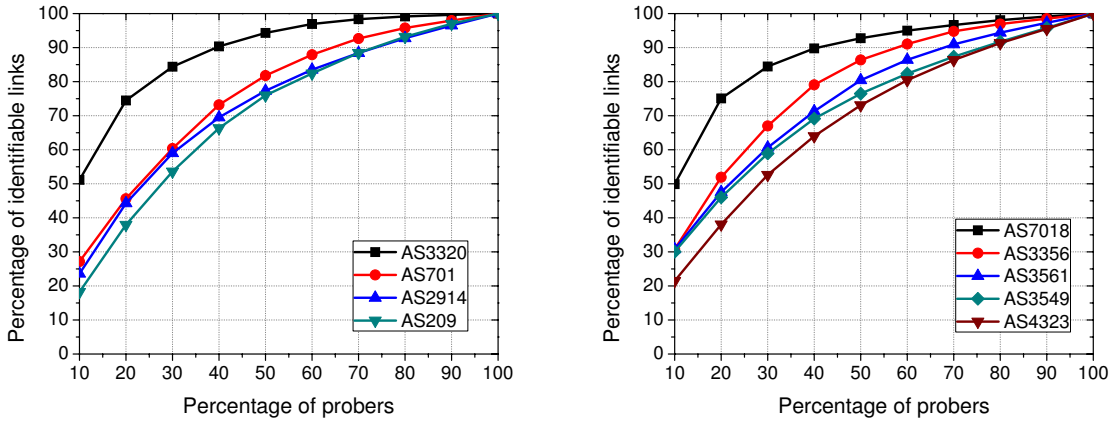
2. *Cost per identifiable target link*: The algorithm PathSelection first selects probing paths to uniquely determine identifiable target links, and then to cover unidentifiable target links. We count the first part of probing cost and amortize it to identifiable target links. In linear system, n linear equations can solve at most n variables. Therefore, the lower bound of the cost per identifiable target link is 1.

2.4.2 Percentage of Identifiable Links

In this subsection, we investigate how many links are identifiable in a network consisting of symmetrical and asymmetrical links. In the simulation, the cost of an asymmetrical link in each direction is set to a random number. Compared to networks of symmetrical links, the number of linear equations and variables are doubled in networks of asymmetrical links. For each topology, we randomly choose 10% to 100% of routers as probers and compute the percentage of identifiable links. We run each simulation 10,000 times and take the average.

The results on networks with symmetrical links are shown in Fig. 2.5. Using more probers results in a higher percentage of identifiable links. When every node is a prober, the performance of each link can be directly measured, and thus links are all identifiable. In networks of symmetrical links, the percentage of identifiable links is quite high. In all nine topologies, more than 50% of links are identifiable if using using 30% of nodes as probers. In particular, this percentage is as high as 85% in AS3320 and AS7018.

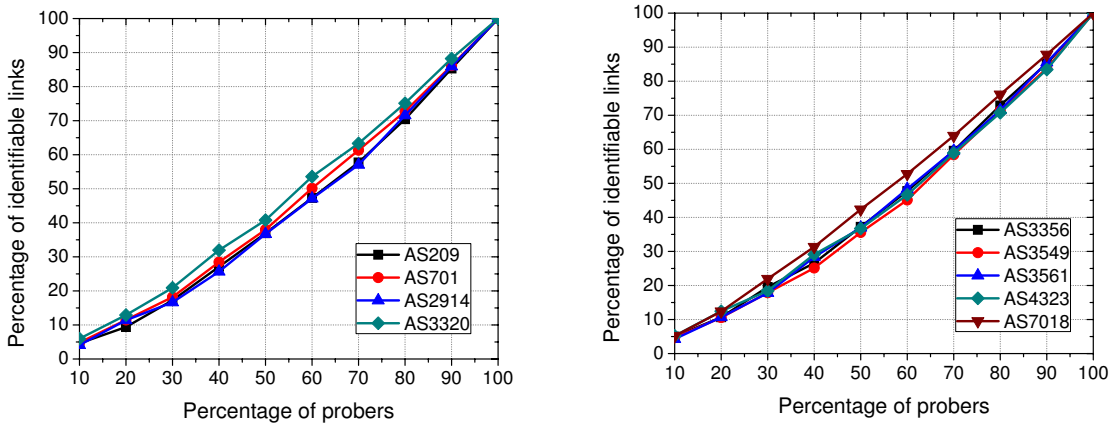
The results on networks with asymmetrical links are shown in Fig. 2.6. The percentage of identifiable links is lower than that in networks with symmetrical links, but it is still high. In summary, the simulation result shows that there are quite many identifiable links even when links are asymmetrical.



(a) AS209, AS701, AS2914, and AS3320

(b) AS3356, AS3549, AS3561, AS4323, and AS7018

Figure 2.5. The percentage of identifiable links under different percentages of probers (links are symmetrical).



(a) AS209, AS701, AS2914, and AS3320

(b) AS3356, AS3549, AS3561, AS4323, and AS7018

Figure 2.6. The percentage of identifiable links under different percentages of probers (links are asymmetrical).

2.4.3 Overall Probing Cost

Next, we evaluate the overall probing cost. Fig. 2.7 shows the overall probing cost when 40% of nodes are probers, which has two features. First, increasing parameter α, i.e., using more solutions for path selection, can reduce the overall probing cost, especially in AS209, AS2914, AS4323, and AS7018. These four topologies have small scales compared to others. Hence, for a small-scale network

we may choose a large α to achieve good performance. For a large-scale network, we can use a small α to significantly reduce the running time but sacrifice a little bit of performance. Second, as the percentage of target links increases, more and more non-redundant probing paths are selected. Accordingly, the overall probing cost gradually converges to the theoretical upper bound shown in Theorem 4. Therefore, when the percentage of target links reaches 100%, the overall probing cost under different α becomes similar. When 20% and 60% of nodes are probers, the overall probing cost has similar trend, and will not be shown here.

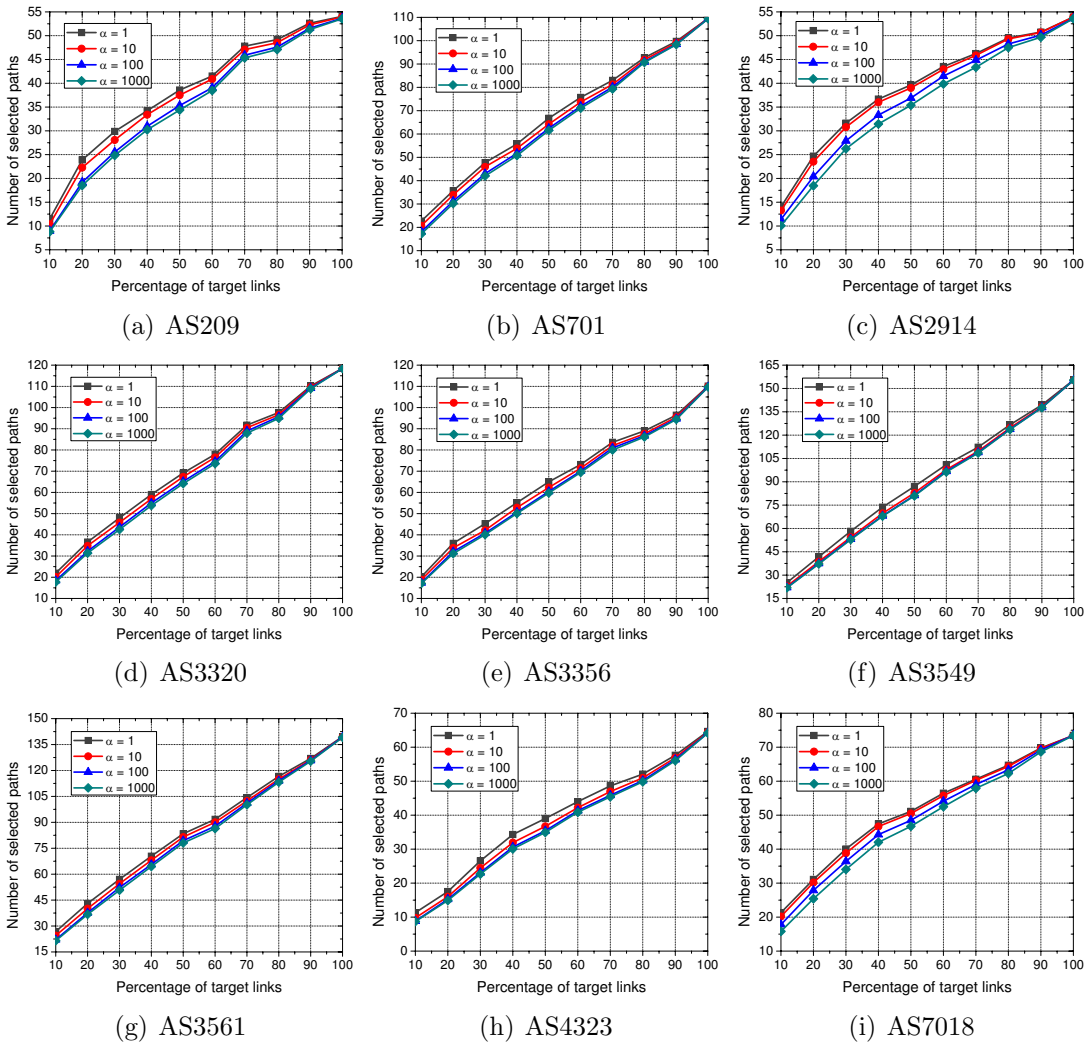


Figure 2.7. The overall probing cost of the path selection algorithm with $\alpha = 1, 10, 100, 1000$ (40% nodes as probers).

2.4.4 Cost Per Identifiable Target Link

Fig. 2.8 shows the cost per identifiable target link with $\alpha = 1000$, when 20%, 40%, and 60% of nodes are probers. In all topologies, this cost quickly decreases as the percentage of target links increases. When the percentage of target links reaches 100%, the cost per identifiable target link is very close to the lower bound 1. This indicates that our path selection algorithm can effectively eliminate redundant probing paths and choose probing paths that are the most useful for determining multiple identifiable target links. The figure also shows that using more nodes as probers is useful for reducing the cost. Given n probers, we have $\frac{n(n-1)}{2}$ usable probing paths. Hence, we have much more usable probing paths when the percentage of probers increases. As a result, our algorithm can pick out better probing paths to reduce this amortized cost. When α is 1, 10, and 100, the cost per identifiable target link has similar trend, and will not be shown here.

2.4.5 Comparisons

The SelectPath algorithm [50] is the closest work in this area. It can select a minimum set of probing paths to uniquely determine all identifiable links and cover all unidentifiable links in the network. However, this algorithm is only a special case of what our algorithm can do, i.e., when all links are target links. Even for this special case, we show that the performance of our solution is guaranteed to be better than or equal to theirs.

Since the SelectPath algorithm cannot be modified for an arbitrary set of target links, we only compare the performance on the case that all links are target links, although this is not fair to our algorithm which is more general. As shown in [50], the overall probing cost of the SelectPath algorithm is equal to the row rank of the dependency matrix. According to Theorem 4, the row rank of the dependency matrix is the upper bound of the overall probing cost of our algorithm. As a result, our method is theoretically not worse than the SelectPath algorithm.

In addition to the above theoretical result, we also use simulations to compare the overall probing cost when all links are target links. In all 3 (percentages of probers) $\times$ 9 (topologies) $\times$ 4 (α) $\times$ 100 (runs) = 10,800 runs of simulations, there is not a single case that the SelectPath algorithm outperforms our algorithm, which

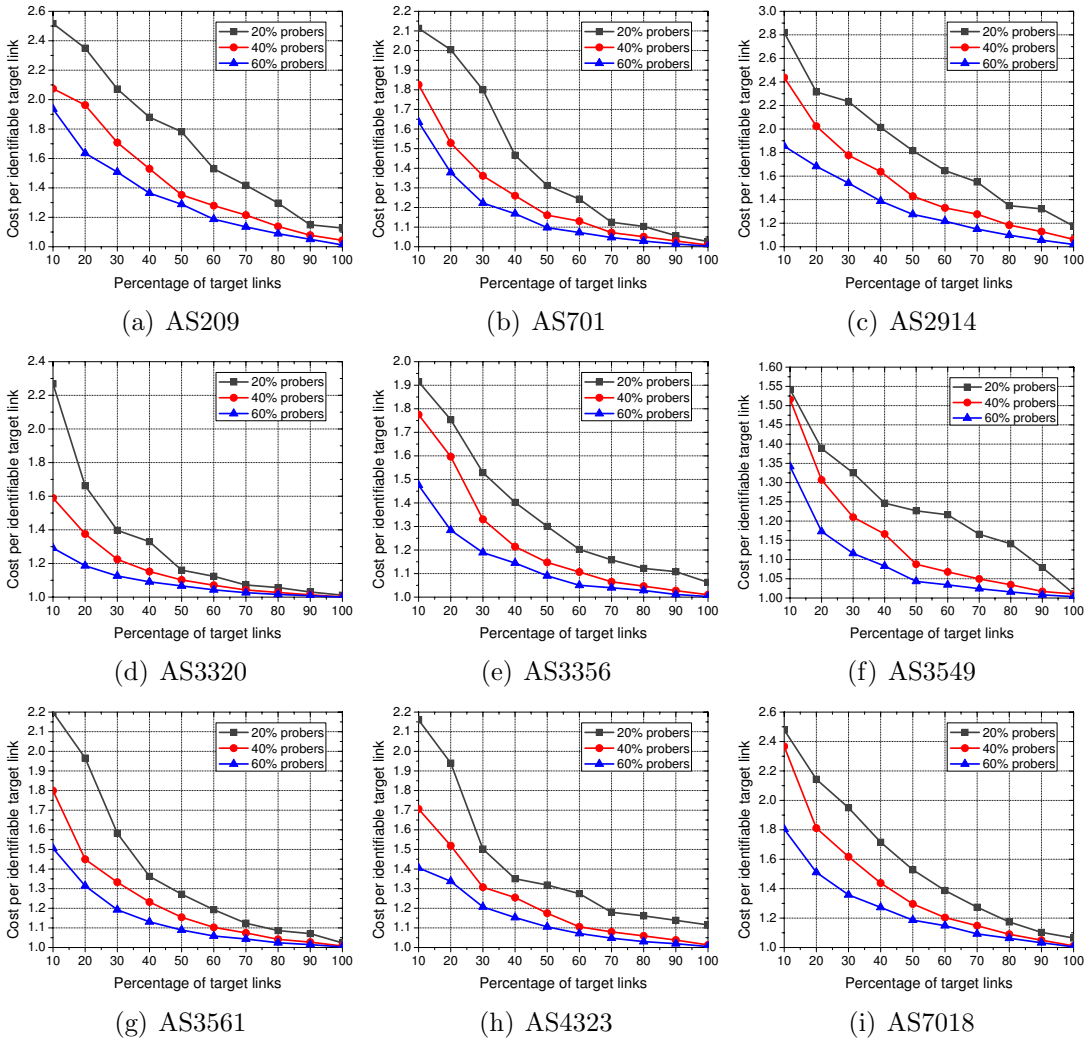


Figure 2.8. The cost per identifiable target link of the path selection algorithm with $\alpha = 1000$. The lower bound is 1.

is consistent with the theoretical result. Table 2.4, Table 2.5, and Table 2.6 show the average overall probing cost of our algorithm with different α and the SelectPath algorithm when 20%, 40%, and 60% of nodes are probers. The simulation result indicates that our algorithm is better than the SelectPath algorithm.

The SelectPath algorithm selects probing paths corresponding to the maximal independent set of row vectors of the dependency matrix. Although vectors in this set are linearly independent, it does not necessarily mean there is no redundancy. Fig. 2.2(b) is a simple example to show this fact. There are three identifiable links e_1, e_2, and e_3. The maximal independent set of row vectors contains 4

Table 2.4. Comparison of overall probing cost of our algorithm and the SelectPath algorithm (percentage of probers: 20%)

Topology	α				SelectPath
	1	**10**	**100**	**1000**	
AS209	23.9	23.7	23.4	23.3	25.4
AS701	50.6	50.5	50.4	50.4	51.6
AS2914	27.9	27.6	27.2	27.1	29.9
AS3320	48.9	48.8	48.8	48.8	49.3
AS3356	47.9	47.6	47.2	47.2	48.8
AS3549	53.2	53.0	53.0	53.0	53.8
AS3561	58.0	58.0	57.8	57.7	59.8
AS4323	26.8	26.8	26.7	26.7	27.1
AS7018	42.8	42.7	42.6	42.6	43.4

Table 2.5. Comparison of overall probing cost of our algorithm and the SelectPath algorithm (percentage of probers: 40%)

Topology	α				SelectPath
	1	**10**	**100**	**1000**	
AS209	52.7	52.6	52.3	52.2	55.7
AS701	107.6	107.6	107.5	107.4	109.0
AS2914	54.2	54.2	53.9	53.9	56.0
AS3320	119.0	119.0	118.9	118.9	119.3
AS3356	107.2	107.0	106.8	106.7	108.6
AS3549	155.0	155.0	155.0	155.0	157.3
AS3561	137.6	137.3	137.2	137.1	140.6
AS4323	64.2	63.9	63.7	63.7	64.8
AS7018	73.6	73.6	73.6	73.6	74.1

vectors. However, we can use only three probing paths p_1, p_2, and p_4 to uniquely determine links e_1, e_2, and e_3. Adding any other probing paths has no contribution to achieving identifiability, no matter it is replaceable by p_1, p_2, and p_4 or not. Our algorithm outperforms the SelectPath algorithm because we try to avoid not only replaceable probing paths but also probing paths without any contribution to achieving identifiability.

Table 2.6. Comparison of overall probing cost of our algorithm and the SelectPath algorithm (percentage of probers: 60%)

Topology	α				SelectPath
	1	**10**	**100**	**1000**	
AS209	74.9	74.8	74.6	74.6	76.7
AS701	149.2	149.2	149.2	149.1	151.0
AS2914	74.8	74.8	74.7	74.7	76.0
AS3320	194.3	194.2	94.2	194.2	194.7
AS3356	174.9	174.9	174.9	174.8	175.7
AS3549	269.2	268.9	268.7	268.7	272.2
AS3561	208.4	208.3	208.1	207.9	210.3
AS4323	99.6	99.4	99.2	99.2	100.6
AS7018	100.3	100.3	100.2	100.2	100.4

2.5 Related Work

Minimizing the probing cost is usually achieved by carefully selecting probing paths. This problem has been studied without resource limitation [21, 50, 28], and in scenarios with explicit operational requirement [27] and resource constraint [60]. Various kinds of probing costs and monitoring objectives have been studied. However, as pointed out in [26], many network inference problems are ill-posed, because the number of measurements are not sufficient to uniquely determine the result. Our work jointly addresses the problem of minimizing the probing cost and achieving identifiability.

Achieving identifiability in network link monitoring is also considered in [22, 48, 49, 50]. However, our work is quite different from theirs. Zhao *et al.* [22] proposed a method for determining the shortest sequence of links whose properties can be uniquely identified by end-to-end probes. It is similar to differentiating the identifiable from unidentifiable links in our work. However, their method does not consider how to minimize the probing cost. The failure localization in [48] aims at accurately pinpointing a failed link, which in essence is to achieve identifiability. However, it only focuses on locating a single link. Our method is able to uniquely determine every identifiable link. Both [49] and [50] deal with the problem of selecting a minimum set of probing paths that can uniquely determine all identifiable links in the network. It is only a special case of what our algorithm can address, i.e., when all links are target links. Even for this special case, the performance

of our solution is guaranteed to be better than or equal to theirs. Another major difference is the method we adopt. Their methods are based on eliminating replaceable paths; i.e., linearly dependent row vectors in the dependency matrix. We take a different approach, and only select probing paths that are the most useful to our objective. In addition to eliminating replaceable probing paths, we also try to avoid selecting probing paths without any contribution to achieving identifiability.

There are some other approaches to infer the delay and loss rate of network links. Nguyen *et al.* [26] exploited the second-order statistics for estimating the loss rate of links. It first infers the variance of the loss rate of links, and then uniquely determines the loss rate of some links with the highest variance. Based on this method, Ghita *et al.* [30] designed an algorithm to minimize the estimation error rate by carefully selecting links with the highest variance of loss rate. Both approaches focus on providing an estimation of loss rate for each link. However, they do not deal with minimizing the probing cost. Additionally, they do not differentiate identifiable links from unidentifiable links. As a result, the estimated loss rate of an identifiable link may be inaccurate. Compared with them, our approach can obtain the accurate loss rate of each identifiable link.

2.6 Conclusions

End-to-end probe has received considerable attention in network link monitoring. We propose an approach for minimizing the probing cost and achieve identifiability. Given a set of target links, the objective is to choose the minimum number of probing paths that can uniquely determine identifiable target links and cover unidentifiable target links. The basic idea is to select probing paths that are the most useful for achieving identifiability and covering unidentifiable target links, and prevent choosing redundant probing paths. Our method eliminates two types of redundant probing paths, i.e., those that can be replaced by others and those without any contribution to achieving identifiability. With our approach, the number of selected probing paths is proved to be bounded. Experiments based on ISP topologies demonstrate that our approach can achieve identifiability with very low probing cost. Compared with prior work, our method is more general and has better performance.

Chapter 3

Detecting and Localizing Large-Scale Router Failures

3.1 Introduction

The current network protection mechanisms, such as IP fast reroute [39], can only deal with sporadic link failures. To recover from large-scale router failures, service providers have to identify the failed routers and then send network operators to repair the network [8]. Consequently, detecting and localizing large-scale router failures are particularly important to enhancing Internet reliability.

The existing works on detecting and localizing network failures only consider small-scale failures [28, 29], and thus are not suitable for large-scale router failures. Some other prior works focus on discovering routing disruption [61, 62], but do not address how to identify the failed routers.

We propose an approach for detecting and localizing large-scale router failures using traceroute-like active probes sent from end systems. Generally, there are two important considerations in active probe-based network failure detection and localization: *probing cost* and *accuracy*. Accordingly, minimizing the probing cost (i.e., the number of probing messages) and accurately localizing the failed routers are two major challenges addressed in our approach.

To minimize the probing cost, our approach consists of a periodic detection phase and a localization phase that is triggered on demand. We carefully choose

probing paths for the two phases. For the detection phase, we aim at using minimal number of probing messages to probe all routers. We formalize this problem as a 0-1 integer programming problem and prove that it is NP-hard. Hence, we propose a greedy algorithm to solve it. For the localization phase, we discover three types of probes that do not provide useful information. We avoid these probes during the localization phase.

For large-scale router failures, active probes may be unable to identify the status of some routers, which will be explained in detail in Section 3.4. We propose a novel distance-based model to estimate the failure probability of those routers. The basic idea is that large-scale router failures are usually within a geographically contiguous area. Hence, a router close to the failed routers may also fail with high probability. Through the estimated failure probability, we can identify routers that are highly likely to have failed.

Experimental results on ISP topologies show that the accuracy of our approach is higher than 96.5%, even when only 10% of routers are connected by end systems for probing. Moreover, the probing cost of our approach is very low and is not affected by the number of end systems used for probing. Compared with prior works, it achieves higher accuracy with much lower probing cost.

The rest of this chapter is organized as follows. In Section 3.2, we present the network model and failure model, and introduce our approach. Section 3.3 and Section 3.4 present the detection phase and the localization phase of the proposed approach. Section 3.5 evaluates the performance of our approach. Finally, Section 3.6 concludes the chapter.

3.2 Overview

In this section, we first introduce the network model and failure model, and then outline our approach.

3.2.1 Network Model and Failure Model

Similar to prior works on IP network monitoring and failure diagnosis [56, 60, 63], we model the network under study as a connected undirected graph $G(V, E)$,

where $V = \{v_i | 1 \leq i \leq n\}$ is the set of nodes (IP routers) and E is the set of edges (bidirectional communication links between IP routers). The failure area is modeled as a geographically contiguous area in the network. Routers within it all fail. We do not make any assumption for the shape and location of the failure area, because in practice the failure area can be anywhere with any shape. Currently, we focus on the scenario of one failure area. Fig. 3.1 shows an example of large-scale router failures.

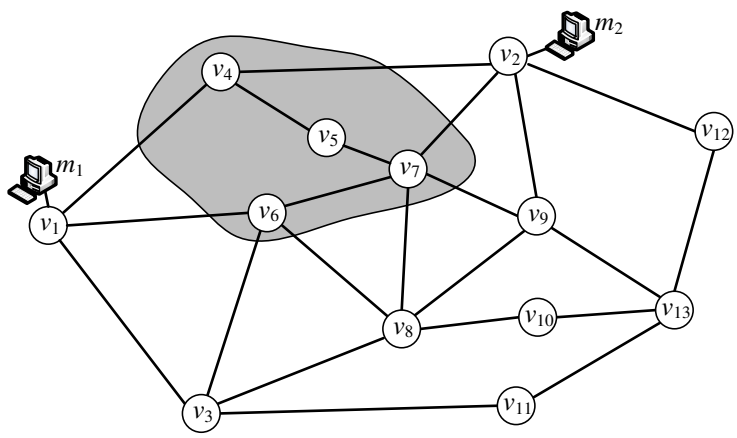


Figure 3.1. An example of large-scale router failures. The shaded region denotes the failure area. Routers within it fail.

The proposed approach is built on a traceroute-like active probe, which is widely used in diagnosing network failures, e.g., [7, 60, 29]. Suppose n_p routers can be directly connected with end systems. We connect an end system with each of them and call these end systems the *probers*. Thus, we have n_p probers $m_1, \cdots, m_{n_p}$. In Fig. 3.1, there are two probers m_1 and m_2. A prober can issue traceroute-like active probes towards all routers. The probe from the prober m_i towards the router v_j follows the routing path $p_{i,j}$ from m_i to v_j. A probing path *covers* a router if it traverses this router.

The central Network Operations Center (NOC) controls probers to issue active probes and collects the probing result from them for failure analysis. When probing a path $p_{i,j}$, the prober sends κ probing messages to each router along $p_{i,j}$ (The default κ in traceroute is 3.). The *probing cost* of $p_{i,j}$ is defined as $\kappa h_{i,j}$, where $h_{i,j}$ is the number of hops in $p_{i,j}$.

When a router receives a probing message, it sends a reply message to the

prober and hence we know that this router is alive. If a probing path contains failed routers, the probe can identify the first failed routers along the path. Additionally, it can determine that the routers between the prober and the first failed router are alive. However, for the routers along the probing path but behind the first failed router, the probe cannot determine whether they are alive or failed. In Fig. 3.1, the probe from m_1 to v_5 follows $m_1 \rightarrow v_1 \rightarrow v_4 \rightarrow v_5$ and the probe from m_1 to v_2 follows $m_1 \rightarrow v_1 \rightarrow v_4 \rightarrow v_2$. They identify that v_1 is alive and v_4 fails. Since v_4 fails, the probing messages cannot reach v_5 and v_2. Here, v_5 fails but v_2 is alive.

Table 3.1. Table of notations

Symbols	**Meaning**
v_j	the jth router in the network
m_i	the ith prober
$p_{i,j}$	routing path from m_i to v_j
$h_{i,j}$	the number of hops in $p_{i,j}$
$a^k_{i,j}$	1 if $p_{i,j}$ traverses v_k, 0 otherwise
$d_{i,j}$	geographic distance between v_i and v_j

3.2.2 Design Overview

Intuitively, probers should periodically probe every path to identify the status of all routers. However, the probing cost is very high. To minimize the probing cost, we divide the whole task into two parts: detecting the occurrence of large-scale router failures and localizing the failed routers. Accordingly, our approach consists of a detection phase and a localization phase. Since we cannot foresee when large-scale router failures occur, the detection phase is periodically invoked to probe all routers. We seek to choose probing paths so as to cover all routers with minimal number of probing messages. The formulation and solution of this problem will be introduced in detail in Section 3.3.

If the detection phase discovers that the number of failed routers exceeds the predefined threshold, the localization phase is immediately triggered to identify all failed routers. We may need to probe some additional paths to check the status of routers. We discover three types of probes that do not provide useful information, and avoid these probes during the localization phase. It is possible that the status of some routers cannot be identified by probes. Hence, we develop a distance-based

model to estimate the failure probability of these routers. Then we determine if they have failed based on the failure probability. The detail of the localization phase will be presented in Section 3.4.

3.3 The Detection Phase

We first define the problem and present the problem formulation, and then propose a greedy algorithm to solve it.

3.3.1 Problem Formulation

The objective is to choose probing paths to cover all routers with minimal number of probing messages. We define this problem as follows.

Definition 3 (Problem of probing path selection)**.** *Given the network $G(V, E)$ and the set of probing paths $P = \{p_{i,j} | 1 \leq i \leq n_p, 1 \leq j \leq n\}$, the objective is to select probing paths from P that satisfy: (1) every router is covered; (2) the number of probing messages is minimal.*

Before presenting the problem formulation, we define variable $a^k_{i,j}$ in Eq. (3.1) for the coverage relation.

$$a^k_{i,j} = \begin{cases} 1 & \text{if } p_{i,j} \text{ covers router } v_k \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

We define variable $c_{i,j}$ in Eq. (3.2) to denote that probing path $p_{i,j}$ is chosen.

$$c_{i,j} = \begin{cases} 1 & \text{if } p_{i,j} \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Our problem can be formalized as a 0-1 integer programming problem as in Eq. (3.3)–Eq. (3.5).

$$\text{minimize} \quad \sum_{i=1}^{n_p} \sum_{j=1}^{n} \kappa c_{i,j} h_{i,j} \tag{3.3}$$

subject to

$$\sum_{i=1}^{n_p}\sum_{j=1}^{n} c_{i,j}a_{i,j}^{k} \geq 1 \tag{3.4}$$

$$1 \leq i \leq n_p, \quad 1 \leq j \leq n, \quad 1 \leq k \leq n, \quad c_{i,j} \in \{0,1\} \tag{3.5}$$

The objective function Eq. (3.3) minimizes the number of probing messages and the constraint in Eq. (3.4) means that the selected probing paths must cover every router.

3.3.2 The Bipartite Model and Greedy Algorithm

We model the above 0-1 integer programming problem with a bipartite $G_B = \{P, V, A\}$. Each vertex in the upper part P and lower part V denotes a probing path and a router. The vertex of $p_{i,j}$ has the attribute $\kappa h_{i,j}$, which is the probing cost of $p_{i,j}$. Here, we set κ to 3, i.e., the default value in traceroute. The set $A = \{a_{i,j}^{k} | p_{i,j} \in P, v_k \in V\}$ denotes the edges between two parts of the bipartite. Fig. 3.2 shows the bipartite model for the example in Fig. 3.1. There are $2 \times 13 = 26$ probing paths in the upper part and we only show 5 of them.

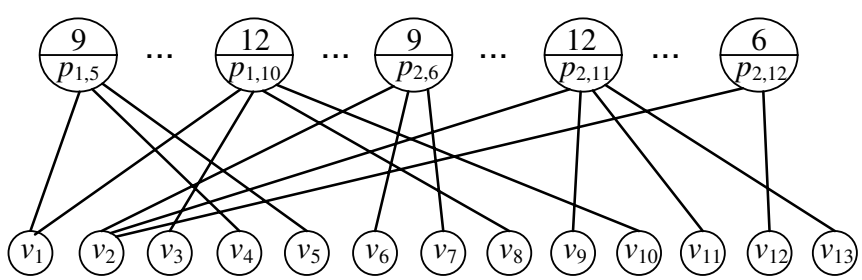


Figure 3.2. The bipartite model of the problem of probing path selection. In each vertex of the upper part, the value above $p_{i,j}$ denotes the attribute $\kappa h_{i,j}$, where κ is set to 3.

With the bipartite model, our problem can be stated as: selecting a set of vertices P_{det} from the upper part such that: (1) every vertex in the lower part has neighbors in P_{det}; (2) $\sum_{p_{i,j} \in P_{det}} \kappa h_{i,j}$ is minimal.

Theorem 5. *The problem of probing path selection is NP-hard.*

Proof. By setting the attribute $\kappa h_{i,j}$ to 1 for every probing path $p_{i,j}$, the problem of probing path selection is the same as the classic set cover problem. This means that the set cover problem is a special case of our problem. Since the set cover problem is NP-hard, our problem is also NP-hard. □

We propose a greedy algorithm PathSelection shown in Algorithm 3 to solve our problem. The algorithm repeatedly selects a vertex from P and removes the corresponding vertices from V, until V becomes empty. Each time, it chooses $p_{i,j}$ with the smallest $\frac{\kappa h_{i,j}}{N_{i,j}}$ (line 4), where $N_{i,j}$, the number of neighbors that are currently uncovered, is computed in line 3. If multiple vertices have the same smallest $\frac{\kappa h_{i,j}}{N_{i,j}}$, we choose $p_{i,j}$ such that v_j is the leaf node of the routing tree of m_i. It helps reduce the probing cost of the localization phase, which will be explained in detail in Section 3.4.2. If the destination of these probing paths with the same smallest $\frac{\kappa h_{i,j}}{N_{i,j}}$ are all non-leaf nodes, we randomly choose a vertex from the candidates. Then, we remove $p_{i,j}$ from P (line 5) and remove the neighbors of $p_{i,j}$ from V (line 7–9). Because of removing $p_{i,j}$ from V, some vertices in P may have no neighbors; thus we remove them from P (line 10–12). When the algorithm terminates, the set P_{det} contains the selected probing paths.

Algorithm 3 PathSelection

Input: The bipartite $G_B = \{P, V, A\}$
Output: The set of probing paths P_{det}
Procedure:

1: $P_{det} \leftarrow \emptyset$
2: **while** $V \neq \emptyset$ **do**
3: $\quad N_{i,j} \leftarrow$ the number of neighbors of vertex $p_{i,j} \in P$
4: $\quad p_{i,j} \leftarrow$ the vertex in P with the smallest $\frac{\kappa h_{i,j}}{N_{i,j}}$
5: $\quad P \leftarrow P - p_{i,j}$
6: $\quad P_{det} \leftarrow P_{det} \cup p_{i,j}$
7: $\quad$ **for** each neighbor v_k of $p_{i,j}$ **do**
8: $\quad\quad V \leftarrow V - v_k$
9: $\quad$ **end for**
10: $\quad$ **for** each vertex $p_{a,b} \in P$ with no neighbors **do**
11: $\quad\quad P \leftarrow P - p_{a,b}$
12: $\quad$ **end for**
13: **end while**

The algorithm PathSelection is invoked when our approach is first deployed and

when the network topology or routing configuration changes. In each detection round, probers issue active probes along the selected probing paths and then send the probing result to the NOC.

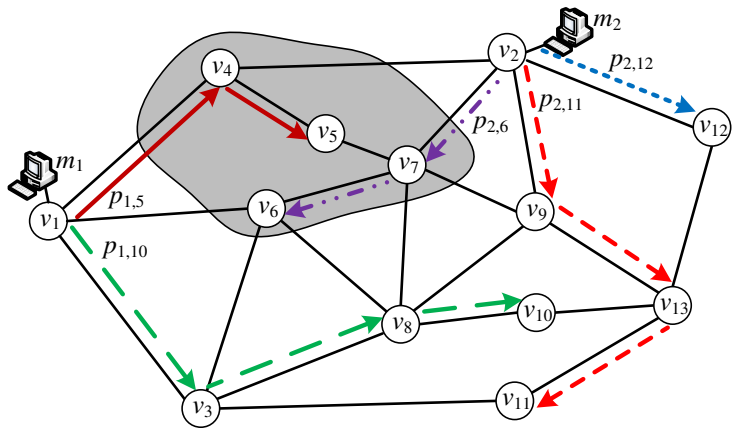


Figure 3.3. The probing paths selected for the detection phase.

For the example in Fig. 3.1, the algorithm PathSelection chooses 5 probing paths $p_{1,5}$, $p_{1,10}$, $p_{2,6}$, $p_{2,11}$, and $p_{2,12}$ as shown in Fig. 3.3. Probing these paths classifies the status of routers into three categories, i.e., *live*, *failed*, and *unknown*, as shown in Table 3.2.

Table 3.2. The probing result of the detection phase

Status	**Routers**
live	v_1, v_2, v_3, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}
failed	v_4, v_7
unknown	v_5, v_6

3.4 The Localization Phase

We first discuss when to trigger the localization phase, and then describe three types of useless probes. Finally, we propose a distance-based model for estimating the failure probability of routers.

3.4.1 Triggering the Localization Phase

We use a simple and practical criterion to determine if the localization phase should be triggered. The localization phase is triggered if the detection phase discovers that the number of failed routers exceeds the predefined threshold τ. This criterion is based on the fact that most failures in the Internet are sporadic link failures [3]. If several router failures are detected, it is likely that large-scale router failures have occurred. We set the threshold τ to 2 in this chapter. As shown in Table 3.2, the detection phase discovers that at least two routers have failed. Hence, the localization phase is triggered.

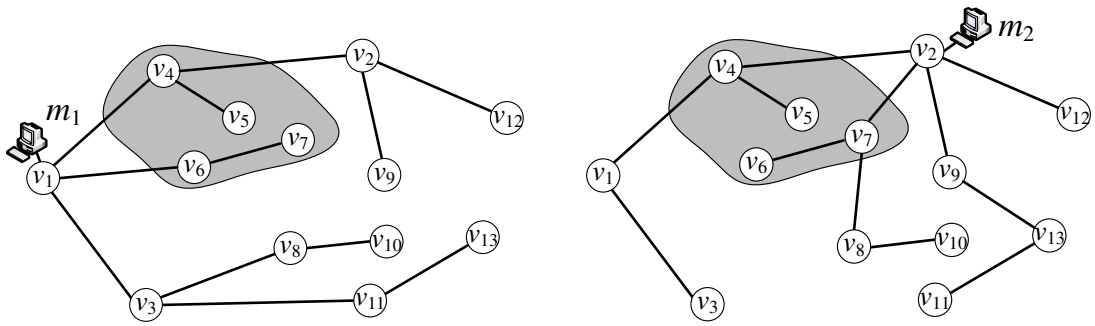
3.4.2 Avoiding Useless Probes

Intuitively, each prober needs to probe paths towards all routers to identify their status. In Fig. 3.4, m_1 needs to issue 13 probes, and so does m_2. We identify the following three types of useless probes. Avoiding them can save many probing messages.

1. The prober only needs to probe the paths towards the leaf nodes of its routing tree, because the paths towards non-leaf nodes are included in the paths towards the leaf nodes. In Fig. 3.4(a), probing paths $p_{1,3}$ and $p_{1,11}$ are a part of $p_{1,13}$, and hence we do not need to probe them.

2. We do not need to probe the paths that are probed in the current detection phase. Unlike temporary link failures, large-scale router failures are usually long lasting. Thus, we can make use of the probing result in the current detection phase. This is the reason that we prefer the probing paths towards the leaf nodes of routing trees in Algorithm 3.

3. For a probing path $p_{i,j}$ containing a failed router v_k, if the routers between m_i and v_k are all live, we do not need to probe $p_{i,j}$, because probing it can only identify the failed router v_k. In our example, the detection phase identifies that v_4 and v_7 fail. As shown in Fig. 3.4, probing paths $p_{1,5}$, $p_{1,9}$, $p_{1,12}$, $p_{2,3}$, $p_{2,5}$, $p_{2,6}$, and $p_{2,10}$ satisfy the requirement, and thus we do not need to probe them.



(a) The routing tree of m_1. (b) The routing tree of m_2.

Figure 3.4. The routing tree of the two probers.

After excluding the above three types of useless probes, the localization phase needs to probe only two paths $p_{1,7} : m_1 \rightarrow v_1 \rightarrow v_6 \rightarrow v_7$ and $p_{1,13} : m_1 \rightarrow v_1 \rightarrow v_3 \rightarrow v_{11} \rightarrow v_{13}$ with $(3+4)\kappa = 7\kappa$ probing messages. Therefore, excluding useless probes can save $\frac{26-2}{26} = 92.3\%$ probes and $\frac{74\kappa-7\kappa}{74\kappa} = 90.5\%$ probing messages. From the probing result of the two phases, we can obtain the status of routers as shown in Table 3.3.

Table 3.3. The status of routers identified by the probes in two phases

Status	**Routers**
live	v_1, v_2, v_3, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}
failed	v_4, v_6, v_7
unknown	v_5

3.4.3 Distance-Based Router Failure Probability Estimation

It is possible that probes cannot identify the status of some routers (e.g., v_5 in Table 3.3), especially when probers are very few or the failure area is large. We propose a distance-based model to estimate the failure probability of the routers with unknown status. Since large-scale router failures are usually in a geographically contiguous area, a router close to the failed routers may also fail with high probability. The basic idea is to map the distance to a failure probability.

Let $d_{i,j}$ be the geographic distance between routers v_i and v_j. The probes in two phases identify some failed routers and the routers with unknown status, which are

denoted by the set V_F and V_U. For $v_i \in V_U$ and $v_j \in V_F$, $P(v_i|v_j)$ is the conditional failure probability of v_i, when we know v_j has failed. We compute $P(v_i|v_j)$ with Eq. (3.6) by mapping $d_{i,j}$ to a failure probability with a function $\mathcal{F}$.

$$P(v_i|v_j) = \mathcal{F}(d_{i,j}) \tag{3.6}$$

There are two requirements for the mapping function $\mathcal{F}$. First, it needs to map $d_{i,j}$ to a real number between 0 and 1, i.e., $\mathcal{F} : \mathbb{R}^+ \to (0, 1)$. Second, it should be a strictly decreasing function, and thus a larger distance is mapped to a smaller failure probability. Many functions satisfy these requirements and can be chosen as $\mathcal{F}$. We compared several functions and choose $\mathcal{F}(x) = 0.9^{\sqrt{x}}$ in this chapter.

Based on the conditional failure probability $P(v_i|v_j)$, we calculate the failure probability of v_i with Eq. (3.7).

$$f_i = 1 - \prod_{v_j \in V_F} (1 - P(v_i|v_j))) \tag{3.7}$$

In our example, the probes identify that v_4, v_6, and v_7 fail, and the status of v_5 is unknown. The calculation of the failure probability of v_5 is shown in Fig. 3.5.

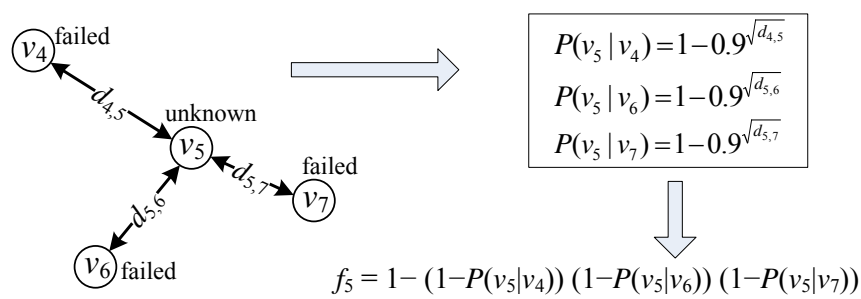


Figure 3.5. The calculation of the failure probability of v_5.

Let $\gamma \in (0, 1)$ be the predefined threshold. A router v_i is said to fail if its estimated failure probability f_i is larger than γ. The threshold γ should be selected according to the mapping function $\mathcal{F}$ and the range of the distance $d_{i,j}$. In Section 3.5, we will investigate the accuracy of our approach with different γ. In summary, the localization algorithm LocalizeFailure is shown in Algorithm 4.

Algorithm 4 LocalizeFailure

Input: The set V_F and V_U
Output: The set V_F
Procedure:

```
for each v_i ∈ V_U do
    f_i ← 1 − ∏_{v_j ∈ V_F} (1 − F(d_{i,j})))
end for
for each v_i ∈ V_U do
    if f_i > γ
        V_F ← V_F ∪ v_i
end for
```

3.5 Performance Evaluation

In this section, we evaluate the performance of our approach and compare it with prior works [64, 7].

3.5.1 Simulation Setup

The simulation is based on ten ISP topologies derived from the Rocketfuel project [59], which are summarized in Table 3.4. For each topology, we randomly place nodes in a 2000×2000 area. All topologies adopt the shortest path routing calculated based on hop count. To simplify the simulation, the failure area is a circle randomly placed in the 2000×2000 area with the radius randomly selected between 400 and 700. Nodes within the circle all fail.

Table 3.4. Summary of topologies used in simulation

Topology	# Nodes	# Links
AS209	58	108
AS701	83	219
AS1668	53	64
AS2914	70	111
AS3257	41	87
AS3320	70	355
AS3356	63	285
AS3549	61	486
AS3561	92	329
AS4323	51	161

Generally, most routers in the Internet cannot be directly connected with probers. Hence, the percentage of routers connected with probers is varied from 2% to 20% in increments of 2%. We randomly select routers and connect probers with them. The parameter κ is set to 3, i.e., a prober needs three messages to probe a router. The threshold τ used in triggering the localization phase is set to 2. We choose $\mathcal{F}(x) = 0.9^{\sqrt{x}}$ as the mapping function of Algorithm 4. The threshold γ in Algorithm 4 is varied from 0.1 to 1.0 with steps of 0.1. We run each simulation 1,000 times and report the average across the simulation set.

We compare the accuracy and probing cost of our algorithm LocalizeFailure with the prefix probing [64, 7], in which each prober sends probes towards the leaf nodes of its routing tree.

3.5.2 Accuracy

First we investigate the accuracy of our approach. The detection and localization result has three possibilities.

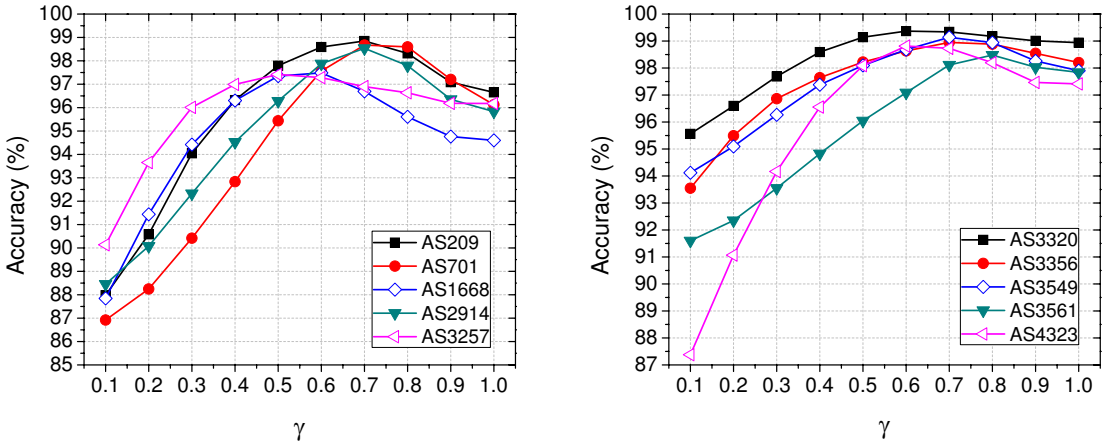
1. A live router is identified as live, or a failed router is identified as failed. Suppose the status of x routers are correctly identified. Then, the *accuracy* is defined as the ratio $\frac{x}{n}$, where n is the total number of routers.

2. A live router is identified as failed. Suppose y routers are in this case. The *false positive rate* is the ratio $\frac{y}{n}$.

3. A failed router is identified as live. Suppose z routers are in this case. The *false negative rate* is the ratio $\frac{z}{n}$.

The accuracy of our approach with varying threshold γ is shown in Fig. 3.6 and Fig. 3.7. We only show the result when 10% and 20% of routers are connected with probers. The highest accuracy is achieved when $\gamma = 0.7$, which is higher than 96.5% in all topologies.

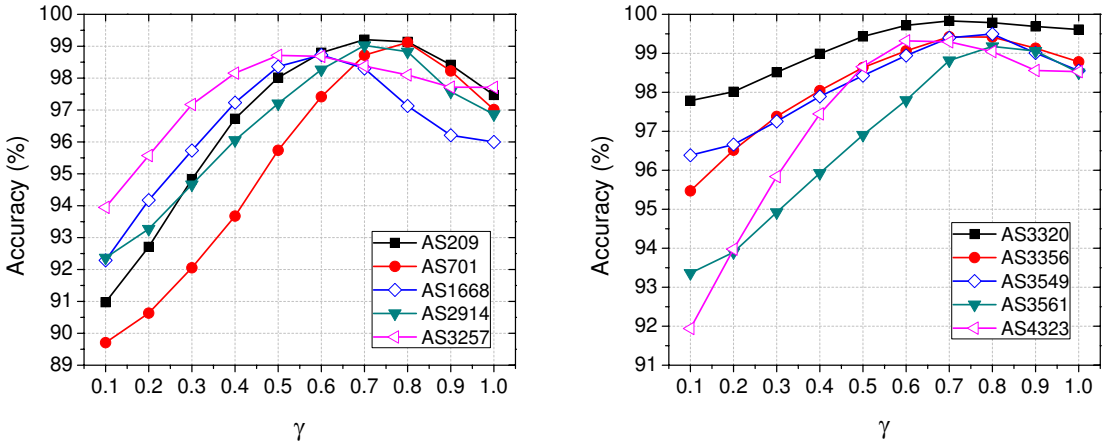
The false positive rate and false negative rate when $\gamma = 0.7$ are shown in Table 3.5. We only show results for three topologies. The high accuracy together with the low false positive rate and false negative rate show the effectiveness of our approach.



(a) AS209, AS701, AS1668, AS2914, and AS3257

(b) AS3320, AS3356, AS3549, AS3561, and AS4323

Figure 3.6. The accuracy when 10% of routers are connected with probers.



(a) AS209, AS701, AS1668, AS2914, and AS3257

(b) AS3320, AS3356, AS3549, AS3561, and AS4323

Figure 3.7. The accuracy when 20% of routers are connected with probers.

We compare our algorithm LocalizeFailure ($\gamma = 0.7$) with the prefix probing and show the result in Fig. 3.8. Our approach achieves much higher accuracy than prefix probing. Active probes may not identify the status of some routers, especially when there are few probers. The prefix probing cannot deal with these routers, while our approach uses the distance-based method to estimate the failure probability of these routers. As a result, our approach achieves higher accuracy and the number of probers has little affect on the accuracy of our approach.

Table 3.5. The false positive rate (%) and false negative rate (%) when $\gamma = 0.7$ (Pos: false positive; Neg: false negative)

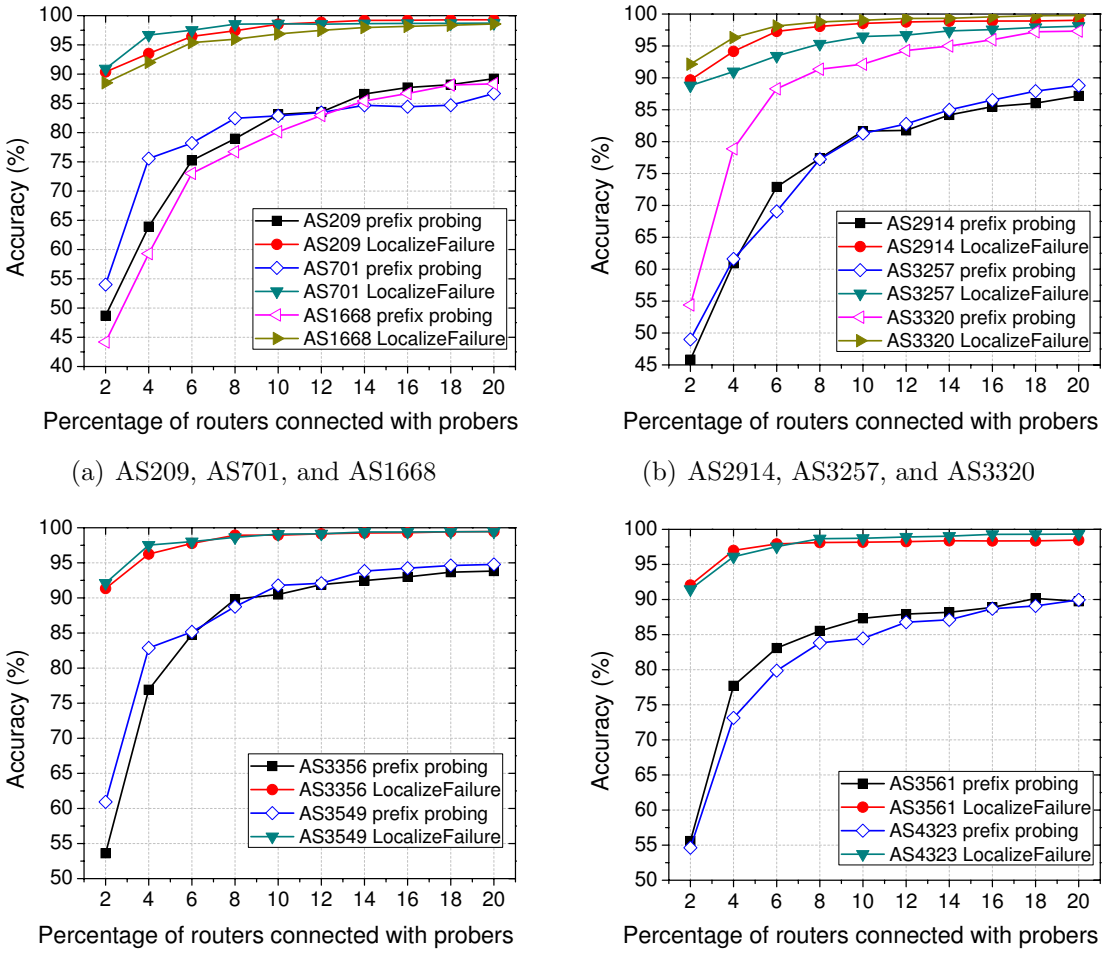
Prober ratio (%)	**AS209**		**AS2914**		**AS3549**	
	Pos	Neg	Pos	Neg	Pos	Neg
6	0.47	3.08	0.69	2.03	0.50	1.48
8	0.44	2.14	0.51	1.42	0.53	0.83
10	0.51	0.96	0.68	0.79	0.66	0.25
12	0.46	0.69	0.54	0.56	0.48	0.39
14	0.46	0.24	0.56	0.53	0.43	0.18

3.5.3 Probing Cost

Next we compare the probing cost of our algorithm LocalizeFailure and that of prefix probing. The performance metric *average probing cost* is defined as the average number of probing messages per router. The probing messages in our approach includes the probing messages in the detection phase and the localization phase. The evaluation result is shown in Fig. 3.9. The average probing cost of our approach is much lower than that of prefix probing. In prefix probing, the average probing cost is proportional to the number of probers, because each prober sends probes towards the leaf nodes of its routing tree. Our approach excludes three types of useless probes. Thus, the average probing cost is very low and increases quite slowly when the number of probers increases. Hence, our method has much better scalability than prefix probing.

3.6 Conclusions

We proposed a two-phase approach for detecting and localizing large-scale router failures. For the detection phase, we propose an algorithm to choose probing paths which can cover all routers with minimal number of probing messages. For the localization phase, we identify three types of useless probes. Avoiding these probes helps reduce the probing cost significantly. For the routers whose status cannot be identified by probes, we develop a distance-based method to estimate their failure probability. Experimental results based on ISP topologies show that the accuracy of our approach is higher than 96.5%, even when only 10% of routers are connected by end systems for probing. Compared with prior works, our method



(a) AS209, AS701, and AS1668

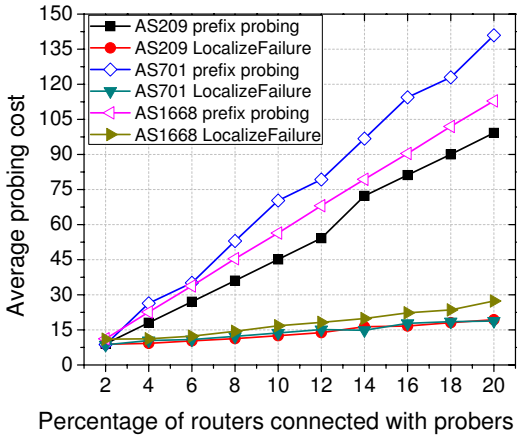
(b) AS2914, AS3257, and AS3320

(c) AS3356 and AS3549

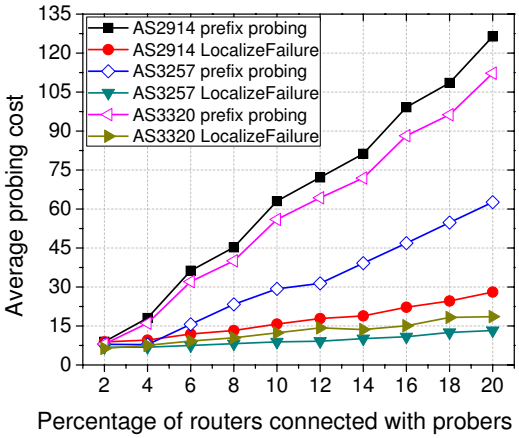
(d) AS3561 and AS4323

Figure 3.8. The accuracy of the prefix probing and the algorithm LocalizeFailure with $\gamma = 0.7$.

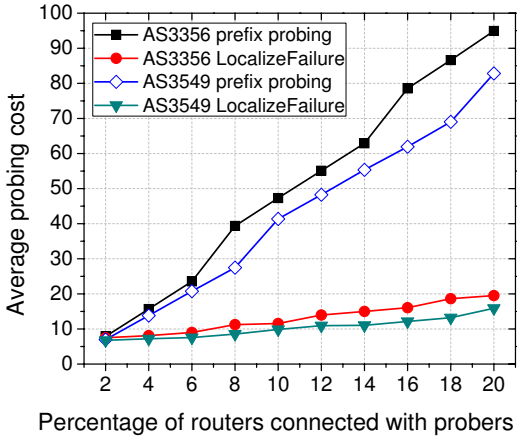
achieves much higher accuracy and with considerably lower probing cost.



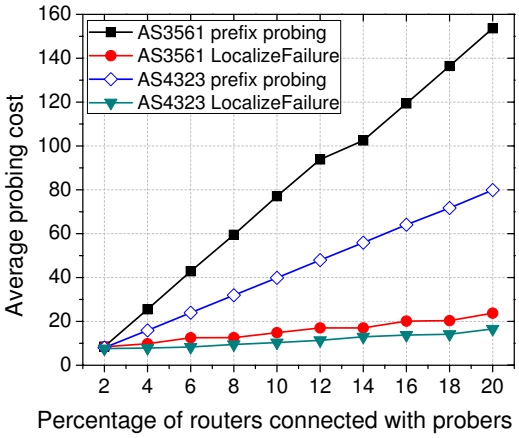
(a) AS209, AS701, and AS1668



(b) AS2914, AS3257, and AS3320



(c) AS3356 and AS3549



(d) AS3561 and AS4323

Figure 3.9. The average probing cost of the prefix probing and the algorithm LocalizeFailure.

Chapter 4

Optimal Recovery from Geographically Correlated Failures

4.1 Introduction

As foundation of the Internet, IP networks are intrinsically robust, but usually need a long time to recover from failures. Interior Gateway Protocols (IGP) for intra-domain routing are designed to recover from failures through updating the routing table of routers. When a router detects a failure, it disseminates topology updates to the other routers in the same Autonomous System (AS). The topology updates trigger these routers to recompute routing paths and update their routing tables. This *IGP convergence* process begins when the failure is detected and finishes after routers update their routing tables. Unfortunately, the IGP convergence is a time consuming process, which usually takes several seconds even for a single link failure [17]. During IGP convergence, a large number of packets may be dropped due to invalid routes. Disconnection of an OC-192 link (10 Gb/s) for 10 seconds can lead to about 12 million packets being dropped on a link (assuming that the average packet size is 1,000 bytes). Under geographically correlated failures, many routers and links within an area fail simultaneously as shown in Fig. 4.1, and thus packet loss would be much more severe.

Many approaches have been proposed to recover disrupted routing during IGP convergence. However, they cannot effectively deal with geographically correlated

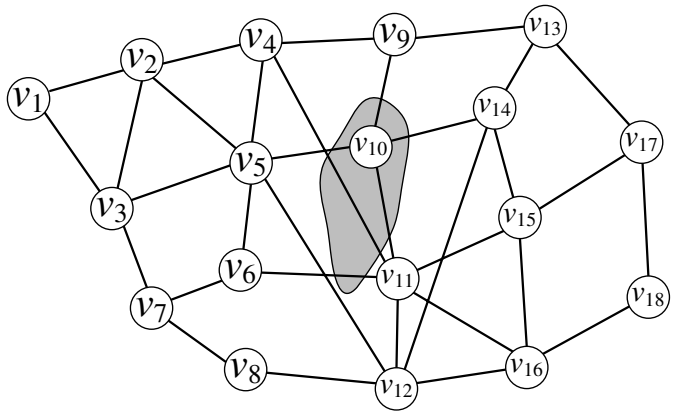


Figure 4.1. An example of geographically correlated failures. The failure area is denoted by shaded area.

failures. A straightforward recovery method is to accelerate the IGP convergence via appropriately tuning the parameters related to failure detection, topology update dissemination, and routing path recomputation [65]. However, rapidly triggering the IGP convergence may cause route flapping and make networks more unstable [66]. Therefore, most existing recovery approaches are proactive and use precomputed backup paths to reroute traffic affected by failures during IGP convergence, e.g., [32, 36, 40]. These proactive approaches are built on an implicit assumption that there are only sporadic and isolated link failures in IP networks. Hence, they are not suitable for geographically correlated failures, because links and their backup paths may fail simultaneously.

Two features make recovery from geographically correlated failures quite difficult. First, the failure area can be of any shape and at any place, and thus we cannot foresee which routers and links will be destroyed. This feature makes proactive recovery approaches unable to effectively deal with geographically correlated failures. Second, no individual router has the overall information of failures, and each router only knows whether its neighbors are reachable. For an unreachable neighbor, the router cannot differentiate whether the neighbor fails or the link connecting the neighbor fails. In Fig. 4.1, the router v_{11} finds that its neighbor v_4, v_6, and v_{10} are unreachable, but it does not know that v_{10} fails while v_4 and v_6 are live. Moreover, v_{11} cannot see that v_{10} is unreachable from v_5.

We propose an approach called Reactive Two-phase Rerouting (RTR) for intra-domain routing to quickly recover from geographically correlated failures with the

shortest recovery paths. The basic idea is to collect failure information, and then calculate valid recovery paths and forward packets over them. To collect failure information, we design a protocol to forward packets around the failure area and record failure information in the packet header. This is a very efficient process that allows us to fully characterize the failed region quickly. After gathering failure information, the second phase of RTR is invoked to calculate new shortest paths to bypass the failure area, and then forward packets along the new shortest paths through source routing.

RTR can deal with geographically correlated failures associated with an area of any shape and location and is free of permanent loops. Under a single link failure, RTR guarantees to recover all failed routing paths with the shortest recovery paths. For any failure area, the recovery paths provided by RTR are guaranteed to be the shortest. Experimental results show that RTR can find the shortest recovery paths for more than 98.6% of the failed routing paths with reachable destinations. Compared with prior works, RTR can find more shortest recovery paths with lower computational and transmission overhead. For the failed routing path with unreachable destinations, RTR can quickly identify them to save network resources. Compared with prior works, RTR can save 83.1% of computation and 75.6% of transmission for irrecoverable failed routing paths.

The rest of this chapter is organized as follows. We describe the system model and our problem in Section 4.2. In Section 4.3, we present the design and properties of RTR. Section 4.4 evaluates the performance of RTR. Section 4.5 reviews related work. Finally, Section 4.6 concludes the chapter.

4.2 Preliminaries

In this section, we first present the network and failure models, briefly introduce the recovery process, and then describe the design objectives.

4.2.1 Network Model and Failure Model

We model the network under study as an undirected graph with a set of nodes (routers) and a set of edges (links). The link from node v_i to node v_j is denoted

by $e_{i,j}$, which has a cost $c_{i,j}$. Links can be asymmetric, i.e., $c_{i,j} \neq c_{j,i}$. We focus on intra-domain routing with a link-state protocol like OSPF and IS-IS. The routing path $p_{i,j}$ from node v_i to node v_j is the shortest path calculated based on the link cost. Its length is the sum of the costs of links along $p_{i,j}$.

In intra-domain routing, every node knows the network topology. We assume that routers have a consistent view of the network topology before geographically correlated failures occur. In addition, each node knows the coordinates of all nodes. RTR uses this information to assist in collecting failure information. This assumption is reasonable for the following reasons. First, in intra-domain routing, the administrator of an AS knows the coordinates of all routers in the AS and can record this information at each router. Second, we aim at providing recovery for high speed IP networks. Routers in such networks are immobile and are rarely moved after being deployed. Third, RTR does not require highly accurate coordinates information of routers and thus does not need special location devices like GPS. When the network administrator moves a router (although this rarely happens in high speed IP networks), the coordinates information stored at routers can also be updated.

We do not modify the mechanisms for failure detection and topology update dissemination in the current IP networks. A router only knows whether its neighbors are reachable, but cannot differentiate between a node failure and a link failure. To prevent route flapping, routers do not immediately disseminate topology updates upon detecting a failure. Thus, routers cannot quickly have the overall information of failures.

To be more practical, we do not make any assumption on the shape and location of the failure area. Similar to prior work [57], the failure area is modeled as a continuous area in the network. Routers within it and links across it all fail. A routing path fails if it contains a failed node or link. A node is *adjacent to* the failure area if it has a link across the failure area.

4.2.2 Recovery Process Overview

RTR is for rerouting traffic affected by failures during IGP convergence. When a router detects that the default next hop towards a destination is unreachable, the

router invokes RTR to recover the routing path towards the destination. After IGP convergence finishes, every live router has a reachable default next hop for each destination in the routing table. RTR is not invoked and the link-state protocol is responsible for routing. Therefore, RTR only operates during IGP convergence for traffic whose routing paths fail. In Fig. 4.1, the routing path from v_7 to v_{17} is $v_7 \rightarrow v_6 \rightarrow v_{11} \rightarrow v_{15} \rightarrow v_{17}$ which is disconnected at $e_{6,11}$. When v_6 detects that v_{11} is unreachable, it invokes RTR to reroute traffic towards v_{17}, until IGP convergence completes.

4.2.3 Objectives

Geographically correlated failures may destroy a large number of routing paths. There are two cases if the source of a failed routing path is live. In the first case, the destination is still reachable from the source. For this case, RTR should find a recovery path as short as possible to reach the destination. This is the goal of existing recovery approaches. In the second case, the destination is unreachable from the source, because it fails or is in a different partition from the source. Packets towards an unreachable destination should be discarded as early as possible. However, this case has not been studied in any prior work. A possible reason is that most existing works focus on sporadic link failures where it is unlikely that there are many unreachable destinations, and hence discarding packets late has little negative impact. Geographically correlated failures may make many destinations unreachable, especially when the network is partitioned by failures. Keeping forwarding packets with unreachable destinations wastes network resources and degrades network performance.

Due to searching on-demand for usable routing paths, reactive recovery unavoidably introduces computational overhead to routers. Routers in high speed IP networks need to forward packets very quickly. Using too much CPU time for recovery computation interferes with forwarding the packets which are not affected by failures. Therefore, a reactive recovery mechanism should have low computational overhead and have as low transmission overhead as possible to save bandwidth.

4.3 RTR: Reactive Two-phase Rerouting

This section describes the design and properties of RTR.

4.3.1 Overview

RTR consists of two phases. The first phase is to collect failure information, i.e., identifying which links have failed. In this phase, packets are forwarded around the failure area, and routers adjacent to the failure area record failed links in the packet header. Then, the second phase calculates a new shortest path based on the collected failure information, and uses source routing to ensure that packets are forwarded along the new shortest path.

In Fig. 4.2, suppose v_7 is the source and v_{17} is the destination. The routing path $v_7 \rightarrow v_6 \rightarrow v_{11} \rightarrow v_{15} \rightarrow v_{17}$ is disconnected at $e_{6,11}$, and thus v_6 invokes RTR for recovery. Node v_6 is referred to as a *recovery initiator*, and a *recovery path* is from the recovery initiator to the destination. In the first phase, starting from v_6, packets towards v_{17} are forwarded around the failure area along the dotted lines. When v_5 receives a packet, it inserts the id of $e_{5,10}$ in the packet header. Similarly, v_9 inserts the id of $e_{9,10}$ in the packet header. After packets return to v_6, v_6 knows that five links have failed by checking the packet header. Then, in the second phase v_6 removes the failed links from the network topology and calculate a new shortest path to reach v_{17}, i.e., the dashed lines. When v_6 receives a packet for v_{17}, it forwards it along the recovery path through source routing. If a node is adjacent to the failure area and its default next hop towards a destination is unreachable, the node is a recovery initiator and it invokes RTR to recover the failed routing path. Multiple recovery initiators independently invoke RTR to recover their failed routing paths.

We use data packets rather than designing a special control message to collect failure information due to two reasons. First, designing a special control message complicates the control plane of routers. Second, if a special control message is used, v_6 has to discard packets towards v_{17} before finishing collection of failure information, because v_6 may not have enough buffer space to store these packets. In RTR, the delay of the packets forwarded to gather failure information is increased, but no packet is dropped. As simulations in Section 4.4 show, the first phase is

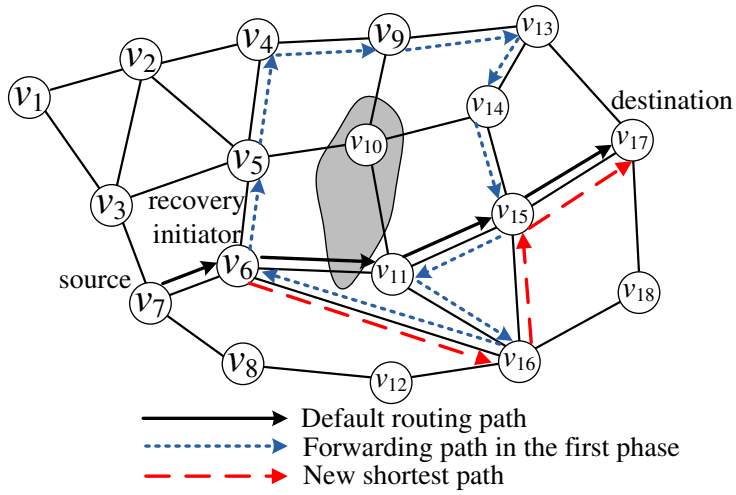


Figure 4.2. The recovery process of RTR on a planar graph.

quite short, and hence the increased delay is small.

The first phase of RTR needs to run only once at a recovery initiator and can benefit all destinations, because failure information can be used for calculating recovery paths for different destinations. The major challenge is to forward packets around the failure area and then back to the recovery initiator.

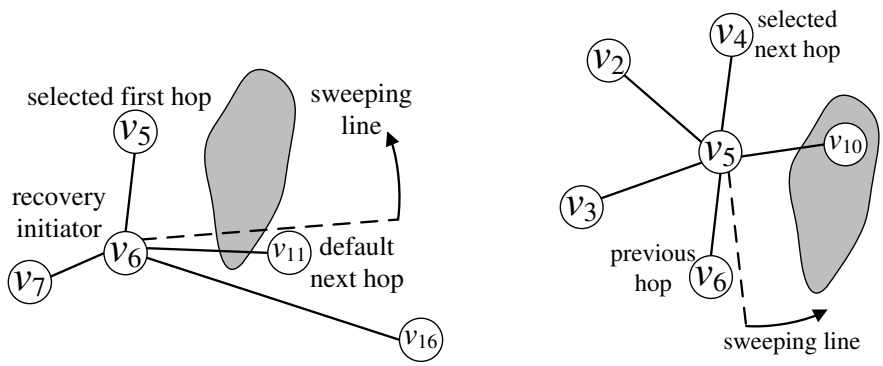
4.3.2 Collecting Failure Information on Planar Graphs

To facilitate exposition, we first explain how to collect failure information on planar graphs with no cross links, and then extend the method to a general graph in the next subsection. Starting from the recovery initiator, we use right-hand rule to select a live neighbor to forward packets. Suppose v_i is the recovery initiator whose default next hop v_j is unreachable, there are two cases.

1. To select the first hop, v_i takes link $e_{i,j}$ as the sweeping line and rotates it counterclockwise, until it reaches a live neighbor. Then, v_i uses this live neighbor as the first hop.

2. In the other cases, suppose v_m receives a packet from v_n, v_m takes link $e_{m,n}$ as the sweeping line and rotates it counterclockwise, until reaching a live neighbor. Then, v_m takes this live neighbor as the next hop.

In Fig. 4.2, the recovery initiator v_6 selects v_5 as the first hop as shown in Fig. 4.3(a), and v_5 chooses v_4 as the next hop as shown in Fig. 4.3(b).



(a) Select the first hop. (b) In the other case.

Figure 4.3. The rule of selecting a live neighbor to forward packets in the first phase.

With the above rule, packets are forwarded out from the recovery initiator v_i to visit every node that is adjacent to the failure area and reachable from v_i, and finally back to v_i, no matter the failure area is inside the network or on the border of the network [67]. To control the forwarding process and collect failure information, we add the following fields in the packet header. The link id is represented by 16 bits.

- `mode`: When it is 0, the packet is forwarded by the default routing protocol. When it is 1, the packet is forwarded by the first phase of RTR.
- `rec_init`: The id of the recovery initiator.
- `failed_link`: It records the *id*s of failed links.

In the first phase, nodes take the following actions.

1. Initially, the recovery initiator v_i sets `mode` in the packet header to 1 and set `rec_init` to v_i. Then, v_i selects the first hop and forwards the packet.
2. When node v_j receives a packet, for each unreachable neighbor v_k other than the recovery initiator v_i, v_j inserts the id of link $e_{j,k}$ in `failed_link` in the packet header if $e_{j,k}$ is not already recorded. Finally, v_j selects the next hop and forwards the packet.

3. When the recovery initiator v_i receives a packet whose `rec_init` is v_i, it selects the next hop. If the next hop is the same as the first hop, it means that the packet has traveled the whole cycle around the failure area, and hence the first phase of RTR completes. Otherwise, v_i forwards the packet to the next hop to prevent missing some nodes on the cycle.

To reduce the transmission overhead, a failed link is not recorded in `failed_link` if v_i is one end of the link, because v_i knows this failure. After the first phase finishes, v_i removes the links in `failed_link` and its links to unreachable neighbors from its view of network topology. Then, v_i calculates the shortest path to the destination and forwards packets with source routing.

Fig. 4.2 shows the recovery process of RTR on a planar graph where v_6 is the recovery initiator and v_{17} is the destination. Along the forwarding path in the first phase, `failed_link` in the packet header records four links $e_{5,10}$, $e_{9,10}$, $e_{14,10}$, and $e_{11,10}$. These links together with $e_{6,11}$ are removed from the network topology when v_6 calculates the shortest path to v_{17}. The new shortest path shown by the dashed lines has 3 hops.

4.3.3 Collecting Failure Information on General Graphs

The forwarding rule introduced above works well on planar graphs, but it may fail to forward packets around the failure area on a general graph as shown in Fig. 4.4. The cycle formed by the dotted lines fails to enclose the failure area. At v_5, the forwarding rule selects v_{12} as the next hop, and thus the dashed cycle fails to enclose the failure area.

This phenomenon looks similar to the permanent loops in geographic routing in wireless networks [68]. However, they are different problems and the solutions in [68] cannot be used to solve our problem. The permanent loop problem in geographic routing refers to the fact that the forwarding rule fails to jump out of local minimum and thus the forwarding path contains a permanent loop. In our problem, the forwarding path fails to enclose the failure area. Unlike geographic routing, we cannot remove some cross links to planarize the topology in advance, because it may incorrectly partition the network when failures occur. For example, the planarization technique may turn the topology in Fig. 4.4 into a planar graph

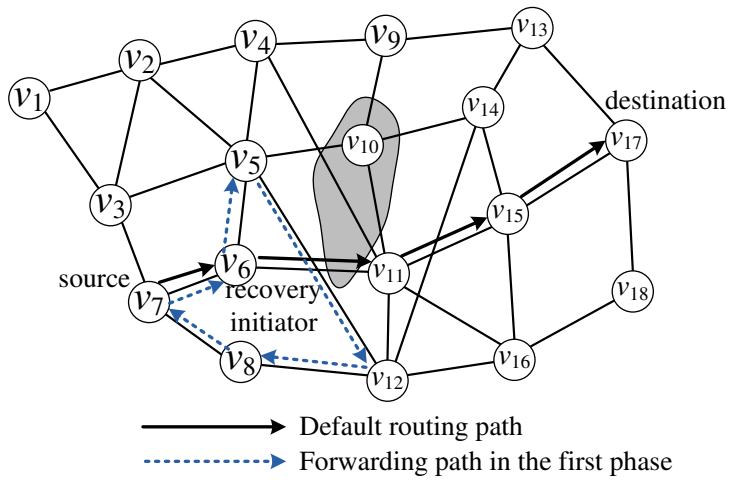


Figure 4.4. The forwarding rule may fail on a general graph.

by removing $e_{6,11}$, $e_{4,11}$ and $e_{12,14}$. The planarized topology is partitioned when $e_{6,5}$ and $e_{6,7}$ fail, but in fact the network is still connected.

We observe that the forwarding rule fails on a general graph because the forwarding path crosses the links between the recovery initiator and its unreachable neighbors. Intuitively, if the forwarding path surrounds the failure area, packets are forwarded clockwise around the failure area as shown in Fig. 4.2. However, when the forwarding path crosses the links between the recovery initiator and its unreachable neighbors, packets are possibly forwarded counterclockwise around the failure area. As a result, the forwarding path fails to enclose the failure area. To solve this problem, we propose the following constraint on the forwarding path.

Constraint 1. *The forwarding path should not cross the links between the recovery initiator and its unreachable neighbors.*

In Fig. 4.4, the forwarding disorder happens at v_5 when it selects v_{12} as the next hop. By Constraint 1, link $e_{6,11}$ prevents $e_{5,12}$ from being selected, and thus v_5 chooses v_4 as the next hop. In this way, packets are forwarded round the failure area.

In addition to the above type of forwarding disorder, another forwarding disorder may also cause problems. As shown in Fig. 4.5, packets traverse a link in each direction, making the forwarding path quite long. When receiving a packet from v_2, v_3 chooses v_4 as the next hop. The selected link $e_{3,4}$ crosses link $e_{1,2}$ that is traversed

before by the forwarding path. This disorder causes the packet to be forwarded counterclockwise around the failure area. If $e_{3,4}$ is excluded, v_3 chooses the correct next hop v_5 and the forwarding path becomes $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_5 \rightarrow v_6 \rightarrow v_1$. Therefore, we have the second constraint on the forwarding path.

Constraint 2. *The forwarding path should not contain cross links.*

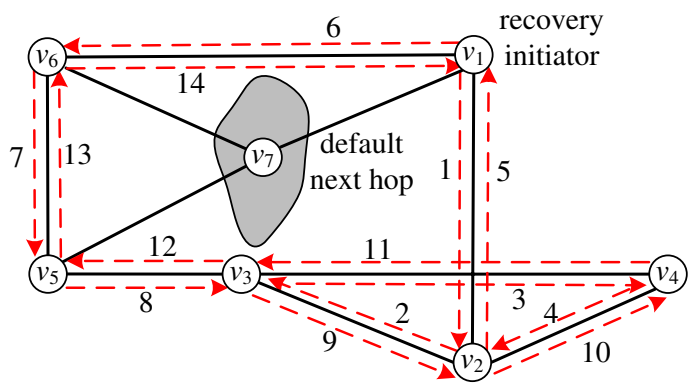


Figure 4.5. The forwarding disorder that leads to a long forwarding path. The numbers beside the dotted lines show the visiting order.

To prevent these two types of forwarding disorders, we add the `cross_link` field in the packet header to record some links. A link across any link in `cross_link` is excluded from being selected to forward packets. For each link, routers precompute the set of links across it. The action taken by each node in the first phase is as follows.

1. Initially, the recovery initiator v_i sets `mode` in the packet header to 1 and set `rec_init` to v_i. For each unreachable neighbor v_j, if link $e_{i,j}$ crosses other links, v_i inserts the id of $e_{i,j}$ in `cross_link`. Then, v_i selects the first hop and forwards the packet.

2. When node v_j receives a packet, for each unreachable neighbor v_k other than the recovery initiator v_i, v_j inserts the id of link $e_{j,k}$ in `failed_link` in the packet header if $e_{j,k}$ is not recorded. Then, v_j selects node v_m as the next hop, where link $e_{j,m}$ is not excluded by the links in `cross_link`. If there is a link which is across $e_{j,m}$ but not excluded by the links in `cross_link`, v_j inserts the id of $e_{j,m}$ in `cross_link`. Finally, v_j forwards the packet to

v_m. (Node v_j is always able to find the next hop, because its previous hop satisfies the requirement.)

3. When the recovery initiator v_i receives a packet whose `rec_init` is v_i, v_i selects the next hop. If it is the same as the first hop, the first phase of RTR ends. Otherwise, v_i forwards the packet to the next hop.

The links recorded in `cross_link` by the recovery initiator are used for preventing the first type of forwarding disorder. All the other links in `cross_link` are used for preventing the second type of forwarding disorder.

Finally, we use an example in Fig. 4.6 to show how RTR works on a general graph. The content of `failed_link` and `cross_link` at each hop is shown in Table 4.1. At v_5, $e_{6,11}$ prevents $e_{5,12}$ from being selected. Similarly, at v_{11}, $e_{14,12}$ blocks $e_{11,15}$ and $e_{11,16}$. The forwarding path in the first phase in shown by the dotted lines. After collecting failure information, v_6 computes a new shortest path towards v_{17} as shown by the dashed lines.

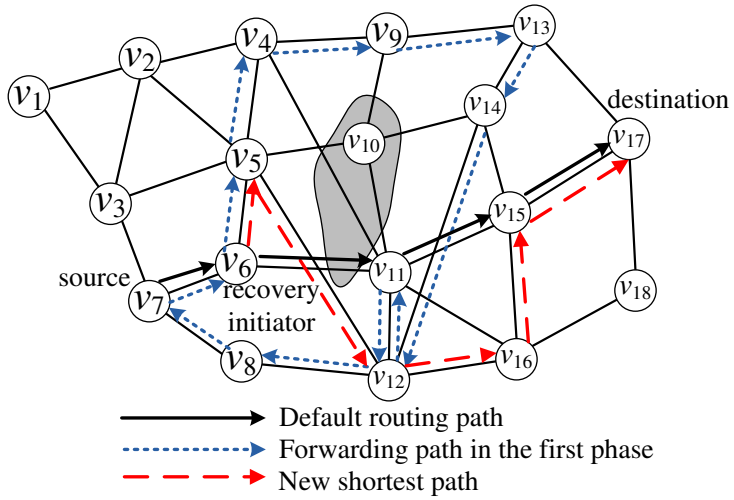


Figure 4.6. The recovery process of RTR on a general graph.

It is possible that the first phase omits some failures. For example, if there is a link between v_{10} and v_{15} in Fig. 4.6, this failed link is omitted. Recording all failed links requires visiting every node that is adjacent to the failure area and reachable from the recovery initiator. This usually leads to a much longer forwarding path and a more complex forwarding rule than the current RTR design. Although the

Table 4.1. The content of the two fields at each hop

Hop count	At node	failed_link	cross_link
0	v_6		$e_{6,11}$
1	v_5	$e_{5,10}$	$e_{6,11}$
2	v_4	$e_{5,10}$, $e_{4,11}$	$e_{6,11}$
3–4	v_9, v_{13}	$e_{5,10}$, $e_{4,11}$, $e_{9,10}$	$e_{6,11}$
5–6	v_{14}, v_{12}	$e_{5,10}$, $e_{4,11}$, $e_{9,10}$, $e_{14,10}$	$e_{6,11}$, $e_{14,12}$
7–11	v_{11}, v_{12}, v_8, v_7, v_6	$e_{5,10}$, $e_{4,11}$, $e_{9,10}$, $e_{14,10}$, $e_{11,10}$	$e_{6,11}$, $e_{14,12}$

first phase does not guarantee to collect all failures, we prove in Section 4.3.5 that RTR can guarantee to find the shortest recovery path to bypass any single link failure as proactive recovery approaches can do. Furthermore, simulations in Section 4.4 show that RTR can find the shortest recovery paths for more than 98.6% of failed routing paths with reachable destinations.

4.3.4 Recomputation and Rerouting

In the second phase, RTR adopts incremental recomputation [69] to calculate the shortest path from the recovery initiator to the destination, which can be achieved within a few milliseconds even for graphs with a thousand nodes [70]. By caching the recovery paths, the recovery initiator needs to calculate the shortest path only once for each destination affected by failures.

After calculating the shortest path, RTR forwards packets along it through source routing, which is widely used in prior works to improve Internet resilience to failures [71, 33, 35, 72]. The recovery initiator inserts the entire shortest path in the packet header. Routers along the shortest path simply forward packets based on the source route in the packet header. Since RTR may omit few failures in the first phase, the recovery path possibly contains a failure. In that case, RTR simply discards the packet.

4.3.5 Properties

This subsection summarizes three major properties of RTR.

Theorem 6. *RTR is free of permanent loops.*

Proof. In the first phase, permanent loops happen when packets cannot return to the recovery initiator. Suppose the forwarding path from the recovery initiator v_i to another node v_{i+k} is $v_i \rightarrow v_{i+1} \rightarrow \cdots \rightarrow v_{i+k}$ consisting of k links. These links do not cross the links between v_i and its unreachable neighbors; otherwise they are excluded from being selected. Moreover, any link across them will not be selected later. It means that these k links will not be excluded from being chosen again. Therefore, in the worst case, packets can be forwarded back to v_i along $v_{i+k} \rightarrow v_{i+k-1} \rightarrow \cdots \rightarrow v_i$. The second phase is free of permanent loops because it forwards packets along the shortest path. □

Next we show the recovery performance and ability.

Theorem 7. *For any failure area, the recovery paths provided by RTR are guaranteed to be the shortest.*

Proof. Let G denote the topology before failures occur. Suppose E_1 is the set of failed links collected by the first phase of RTR, and E_2 is the actual set of failed links (ground truth). We have $E_1 \subseteq E_2$, because RTR may miss failed links but does not label live links as failed. Suppose the routing path from node s to node t is disconnected, and node i is the recovery initiator. Let $p^1_{i,t}$ be the shortest path from node i to node t in $G - E_1$, and $p^2_{i,t}$ be that in $G - E_2$. If RTR successfully recovers the failed routing path from node i to node t, it means that $p^1_{i,t}$ has no failure. Hence, $p^1_{i,t}$ is also a path in $G - E_2$. Since $p^2_{i,t}$ cannot be shorter than $p^1_{i,t}$, $p^1_{i,t}$ is the shortest path from i to t in $G - E_2$. In conclusion, $p^1_{i,t}$ is the shortest recovery path. □

Theorem 8. *For a single link failure, RTR guarantees to recover all failed routing paths with the shortest recovery paths.*

Proof. Theorem 7 shows that the recovery path is the shortest if it does not contain failures. Under any single link failure, the recovery initiator knows the failed link and hence the recovery path does not contain it. Therefore, RTR guarantees to provide the shortest recovery paths. □

Although we focus on a single failure area, RTR also works for multiple failure areas. Upon encountering a failure area F_1, RTR collects failure information of F_1 and computes a backup path to bypass F_1. In order to avoid routing loops, the packet header needs to carry failure information of F_1. When it encounters another failure area F_2, the recovery initiator removes all failed links recorded in the packet header. Through it, the computed recovery path can bypass both F_1 and F_2. To reduce the packet header overhead, we can use the mapping technique in [33] to reduce storage.

4.4 Performance Evaluation

In this section, we evaluate the performance of RTR and compare it with other recovery approaches [32, 33].

4.4.1 Simulation Setup

The simulation is based on eight ISP topologies derived from the Rocketfuel project [59], which are summarized in Table 4.2. For each topology, we randomly place nodes in a 2000×2000 area. All topologies adopt shortest path routing based on hop count. To simplify our simulation, the failure area is a circle randomly placed in the 2000×2000 area with a radius randomly selected between 100 and 300. The radius and location of the circular area are unknown to RTR. Nodes within the circle and links across the circle have all failed. There are three cases for a failed routing path.

1. The source fails and it cannot send packets. We ignore this type of failed routing paths in the simulation.

2. The source is live and the destination is still reachable from the source. This is a *recoverable* failed routing path.

3. The source is live but the destination is unreachable from the source. This is an *irrecoverable* failed routing path.

For a failed routing path with a live source, the recovery process is invoked at the recovery initiator. Some failed routing paths with the same destination may

Table 4.2. Summary of topologies used in simulation

Topology	# Nodes	# Links
AS209	58	108
AS701	83	219
AS1239	52	84
AS3320	70	355
AS3549	61	486
AS3561	92	329
AS4323	51	161
AS7018	115	148

have the same recovery initiator. Their recovery processes are the same; thus we take them as one test case. Given a topology, a test case is determined by three factors, i.e., the recovery initiator, the destination, and the failure area. For each topology, we generate failure areas to collect 10,000 recoverable test cases and 10,000 irrecoverable test cases, and evaluate the performance.

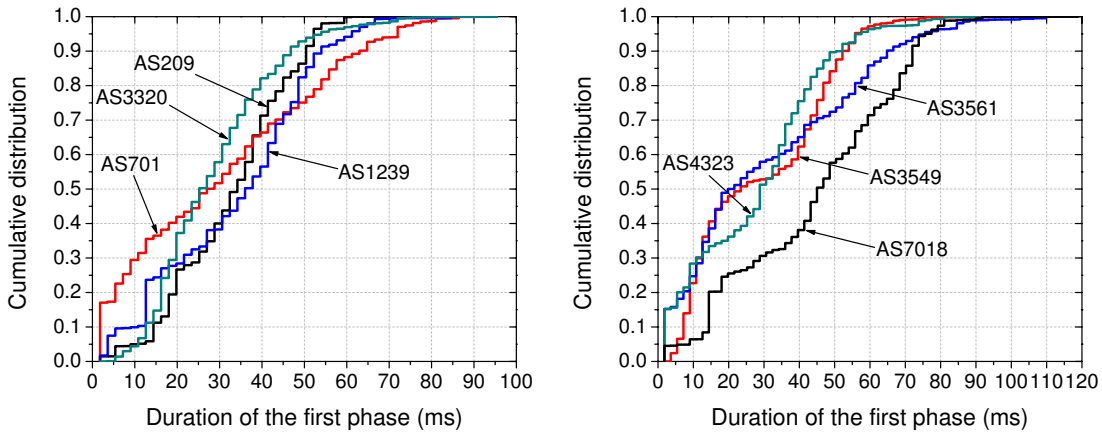
We compare RTR with a proactive recovery approach MRC [32] and a reactive recovery approach FCP [33]. For FCP, we use the source routing version, which reduces the computational overhead of the original FCP.

4.4.2 Duration of the First Phase

The duration of the first phase of RTR affects the delay of packets forwarded during the first phase. RTR has the same first phase in both recoverable and irrecoverable test cases. The one-hop delay consists of the delay at a router and the propagation delay on a link. We use 100 microseconds as the delay at a router, because packets in the first phase introduce quite low computational overhead to routers and 99% of packets in IP networks with OC-12 rate (622 Mb/s) experience less than 100 microseconds delay through a router [73]. The propagation delay on a link is about 1.7 milliseconds, assuming that links are 500 kilometers long on average. Hence, the one-hop delay is 1.8 milliseconds.

The cumulative distribution of the duration of the first phase is shown in Fig. 4.7. The first phase of RTR is quite short. None of the $10 \times 20,000 = 200,000$ test cases has the first phase longer than 110 milliseconds. In every topology, the first phase is shorter than 75 milliseconds in more than 90% of test cases. The

duration in AS7018 is longer than that in the other topologies mainly because this topology has many tree branches. Each link on a tree branch may be traversed twice. Generally speaking, the first phase of RTR is quite short if the topology has high connectivity and few free branches like the Internet backbone.



(a) AS209, AS701, AS1239, and AS3320. (b) AS3549, AS3561, AS4323, and AS7018.

Figure 4.7. The cumulative distribution of duration of the first phase.

4.4.3 Recoverable Test Cases

We evaluate the performance with 10,000 test cases from the following four aspects.

- *Recovery rate*: It is defined as the percentage of successfully recovered test cases. We also measure the optimal recovery rate, i.e., the percentage of successfully recovered test cases with the shortest recovery paths.
- *Stretch*: For a pair of source and destination, the stretch is the ratio of the recovery path length to the length of the shortest path. The optimal value is 1.
- *Computational overhead*: RTR and FCP need to calculate the shortest paths on-demand. The computational overhead is defined as the number of shortest path calculations.
- *Transmission overhead*: RTR and FCP use the packet header to record necessary information for recovery. The transmission overhead is defined as the number of bytes used for recording information.

Table 4.3 summarizes the recovery rate and optimal recovery rate, and Table 4.4 summarizes the maximum stretch and maximum computational overhead. The recovery rate of RTR is higher than 97.7% in every topology. MRC cannot effectively deal with geographically correlated failures, because a routing path and its backup paths may fail simultaneously. Hence, we only compare RTR with FCP in the following part. Theorem 7 shows that if a failed routing path is successfully recovered by RTR, the recovery path is the shortest. Simulation results confirm this property and show that the recovery rate and optimal recovery rate of RTR are the same. RTR achieves higher optimal recovery rate than FCP in every topology. The average optimal recovery rate of RTR is 98.6% in all 100,000 test cases.

Table 4.3. The recovery rate and optimal recovery rate of RTR, FCP, and MRC in recoverable test cases

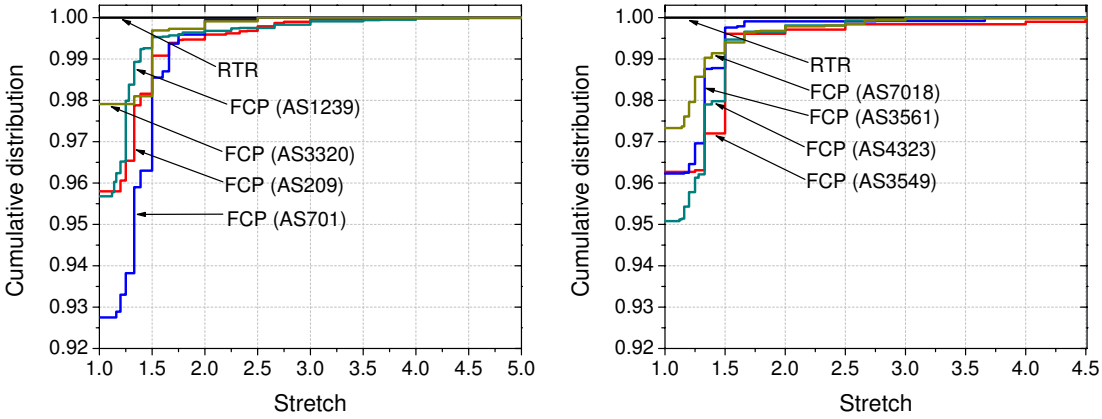
	Recovery rate (%)			**Optimal recovery rate (%)**		
	RTR	FCP	MRC	RTR	FCP	MRC
AS209	98.2	100	36.8	98.2	95.8	13.9
AS701	98.5	100	35.7	98.5	92.8	24.2
AS1239	97.7	100	15.5	97.7	95.7	8.2
AS3320	99.2	100	63.1	99.2	97.9	42.1
AS3549	98.9	100	63.9	98.9	96.3	39.5
AS3561	98.8	100	63.9	98.8	96.2	34.0
AS4323	99.2	100	33.1	99.2	95.1	25.9
AS7018	98.4	100	25.6	98.4	97.3	19.1
Overall	98.6	100	42.2	98.6	95.9	25.9

Next, we compare the stretch of recovery paths provided by RTR and FCP. Fig. 4.8 shows the cumulative distribution of the stretch of successfully recovered paths. RTR guarantees to have optimal stretch 1. FCP achieves small stretch in most test cases, but is still worse than RTR in every topology. As shown in Table 4.4, FCP may have quite a large stretch, which is even worse than MRC.

Next, we compare the computational overhead of RTR and FCP. Fig. 4.9 shows the cumulative distribution of the number of shortest path calculations. In each test case, RTR calculates the shortest path only once, regardless of whether or not it is successfully recovered. FCP uses many more calculations than RTR, because FCP calculates the shortest path whenever the packet encounters a failure not recorded in the packet header.

Table 4.4. The maximum stretch and computational overhead of RTR, FCP, and MRC in recoverable test cases

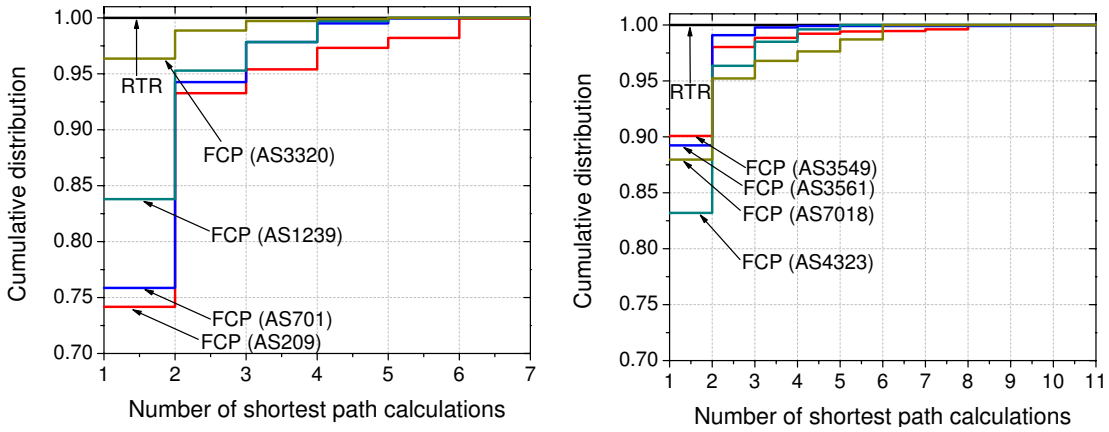
	Maximum stretch			Maximum computational overhead	
	RTR	FCP	MRC	RTR	FCP
AS209	1	4.5	2.5	1	7
AS701	1	2.5	2.5	1	6
AS1239	1	5.0	2.0	1	5
AS3320	1	2.5	3.0	1	5
AS3549	1	4.5	2.0	1	9
AS3561	1	3.7	2.5	1	10
AS4323	1	3.0	2.0	1	5
AS7018	1	4.5	1.8	1	6
Overall	1	5.0	3.0	1	10



(a) AS209, AS701, AS1239, and AS3320. (b) AS3549, AS3561, AS4323, and AS7018.

Figure 4.8. The cumulative distribution of stretch of recovery paths.

Finally, we compare the transmission overhead of RTR and FCP. In the first phase, the transmission overhead of RTR is caused by recording the id of some links. In the second phase, it is due to recording the source route. In FCP, the transmission overhead is from recording the id of encountered failed links and the source route. For each topology, we measure the average transmission overhead across all 10,000 test cases, until IGP convergence finishes. To clearly demonstrate it, we only show the result over the first second in Fig. 4.10. In RTR, the transmission overhead in the first phase is higher than that in the second phase. With the simulation going on, many test cases of RTR enter the second phase, and the transmission overhead quickly decreases. After about 100 milliseconds, all



(a) AS209, AS701, AS1239, and AS3320. (b) AS3549, AS3561, AS4323, and AS7018.

Figure 4.9. The cumulative distribution of computational overhead in recoverable test cases.

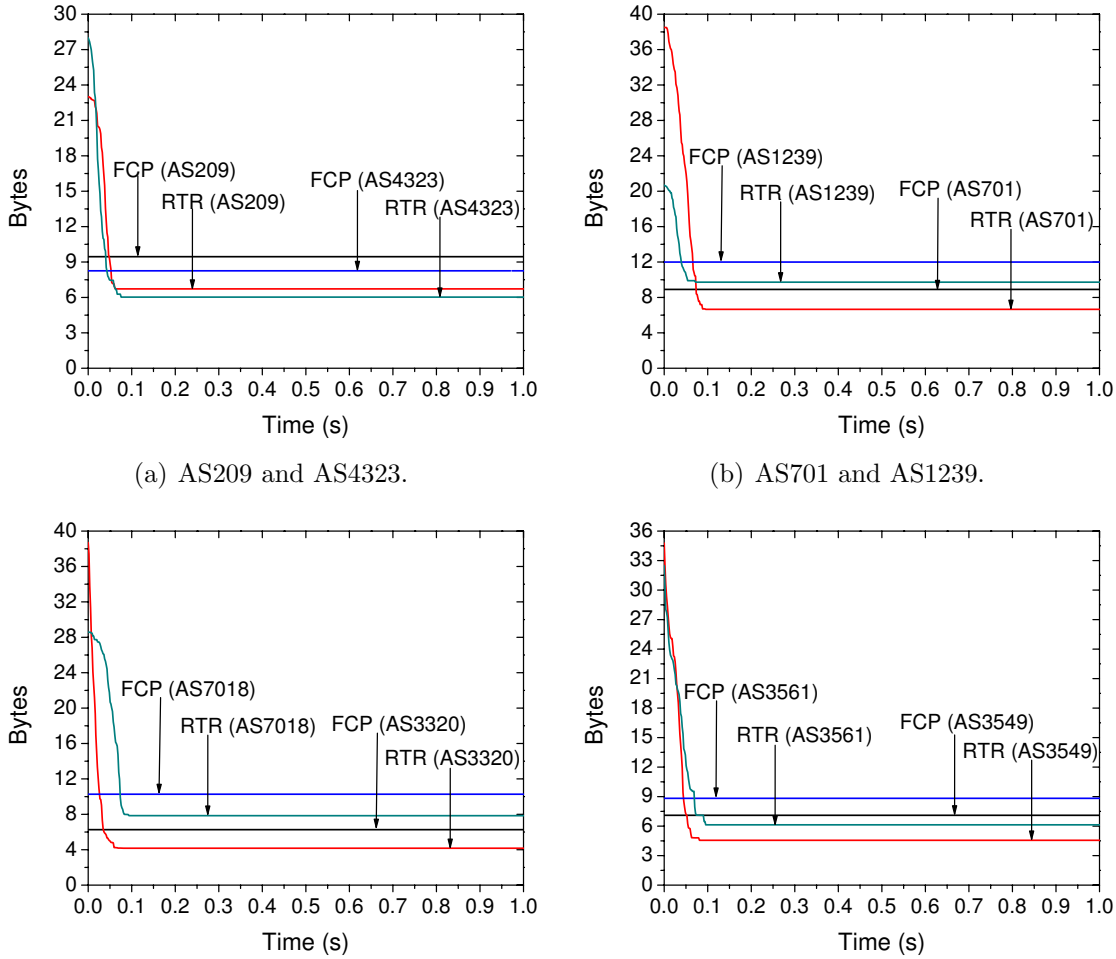
test cases finish the first phase and the transmission overhead converges to a fixed value, which is smaller than that of FCP in every topology.

In summary, the optimal recovery rate of RTR on all 100,000 test cases is as high as 98.6%. Compared with FCP, RTR achieves higher optimal recovery rate with lower computational and transmission overhead.

4.4.4 Irrecoverable Test Cases

First, we investigate the percentage of failed routing paths that are irrecoverable. The radius of the failure area increases from 20 to 300 in increments of 20. For each radius, we generate 1,000 failure areas and count failed routing paths. The percentage of irrecoverable routing paths is shown in Fig. 4.11. Even when the radius is 20 (the failure area takes up about 0.03% of the simulation area), more than 20% of the failed paths are irrecoverable in nine topologies. When the radius is 300 (the failure area takes up about 7.07% of the simulation area), more than 45% of the failed routing paths are irrecoverable in nine topologies. The simulation result illustrates that geographically correlated failures make many failed routing paths irrecoverable. As a result, the performance of a recovery approach on irrecoverable routing paths is also very important.

For an irrecoverable failed routing path, all efforts for recovery are wasted, because packets are ultimately discarded. For each topology, we use 10,000 ir-



(a) AS209 and AS4323.

(b) AS701 and AS1239.

(c) AS3320 and AS7018.

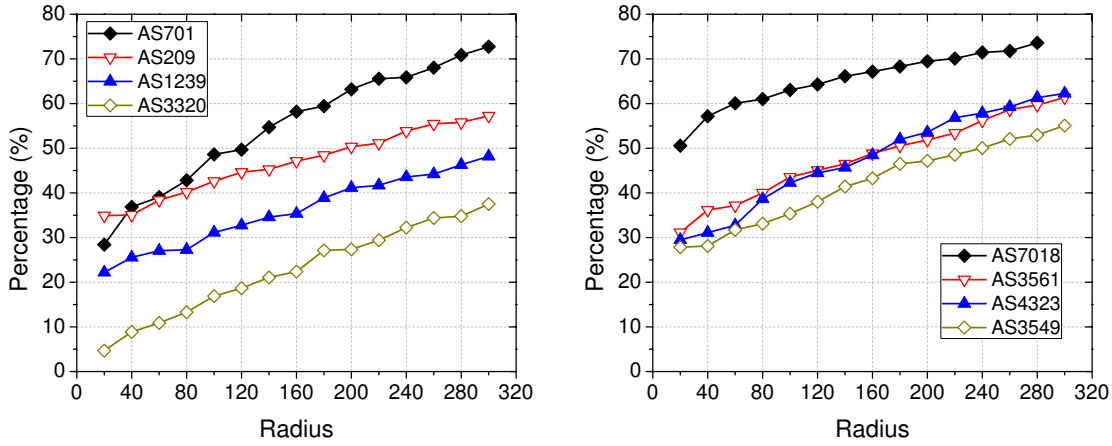
(d) AS3549 and AS3561.

Figure 4.10. The average transmission overhead of RTR and FCP on recoverable test cases.

recoverable test cases to evaluate the performance of RTR and compare it with FCP in the following two aspects.

- *Wasted computation*: It is defined as the number of shortest path calculations.

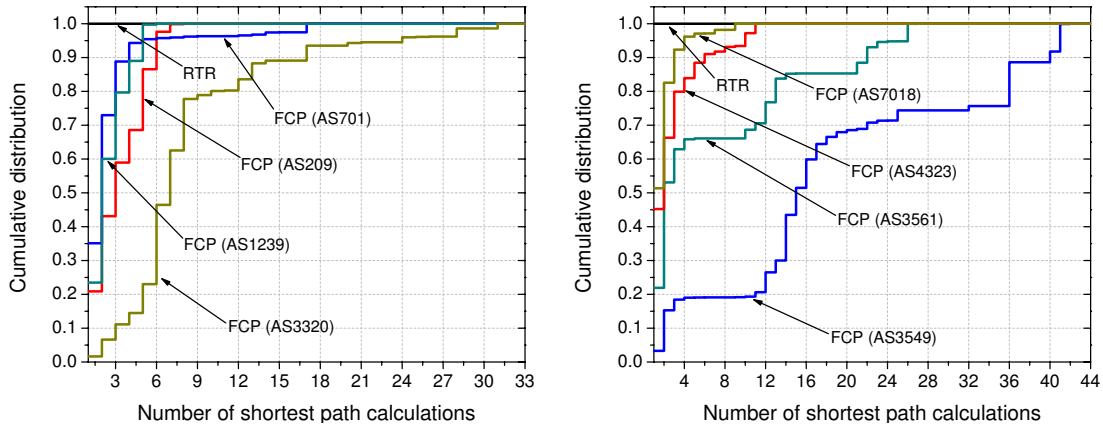
- *Wasted transmission*: We assume that the packet size is 1,000 bytes plus the bytes in the packet header used for recovery. Let s denote this value and h denote the hops from the recovery initiator to the node discarding the packet. Thus, the wasted transmission of the packet is defined as $s \times h$.



(a) AS209, AS701, AS1239, and AS3320. (b) AS3549, AS3561, AS4323, and AS7018.

Figure 4.11. The percentage of failed routing paths that are irrecoverable under the failure area of different radii.

Fig. 4.12 shows the cumulative distribution of wasted computation. The wasted computation of RTR is 1 because it needs to calculate the shortest path only once. The wasted computation of FCP is much more than that of RTR. For example, in AS3549 FCP has more than 10 calculations of the shortest path in about 80% test cases.

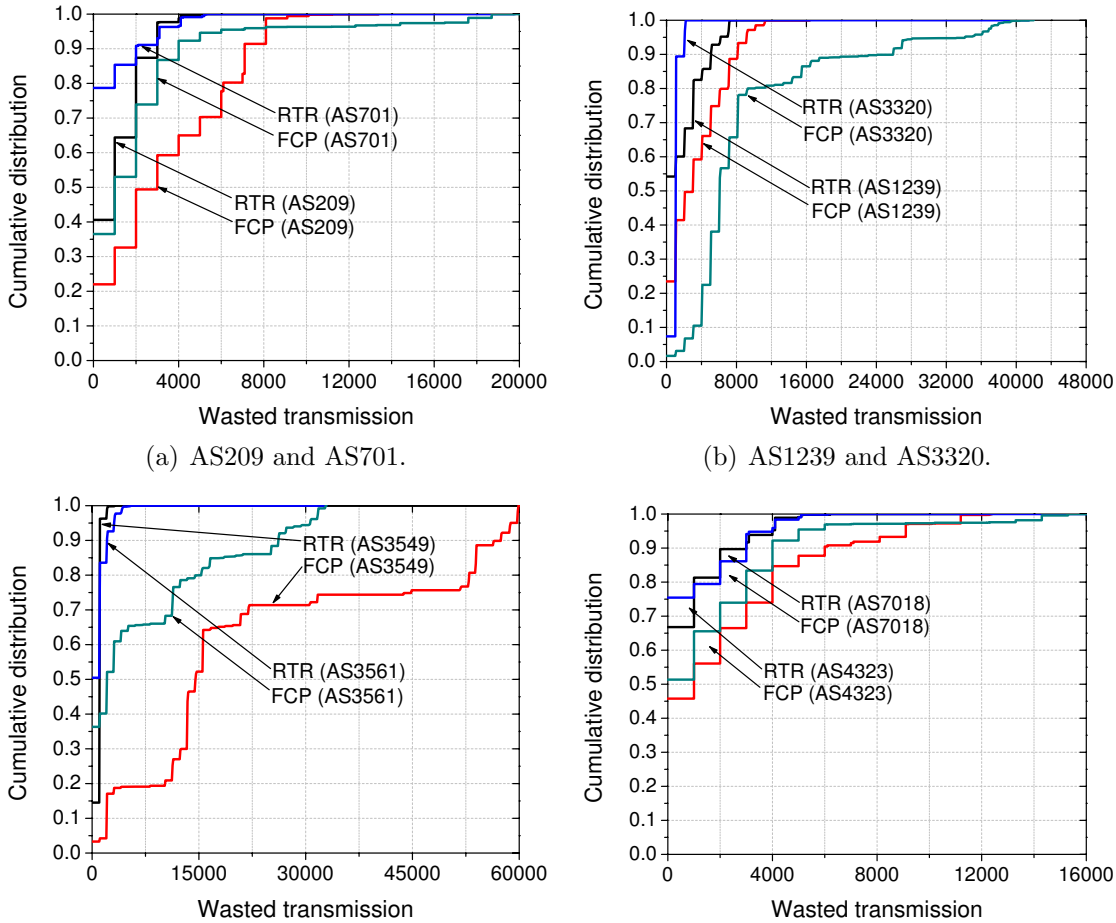


(a) AS209, AS701, AS1239, and AS3320. (b) AS3549, AS3561, AS4323, and AS7018.

Figure 4.12. The cumulative distribution of the wasted computation in irrecoverable test cases.

Next, we compare the wasted transmission of RTR with FCP and show the result in Fig. 4.13. As was the case with the wasted computation, RTR outperforms

FCP in every topology on this metric as well. In particular, the wasted transmission of FCP is much more than that of RTR in AS3320, AS3549 and AS3561.



(a) AS209 and AS701.

(b) AS1239 and AS3320.

(c) AS3549 and AS3561.

(d) AS4323 and AS7018.

Figure 4.13. The cumulative distribution of the wasted transmission on irrecoverable test cases.

Finally, Table 4.5 summarizes the wasted computation and Table 4.6 summarizes the wasted transmission, including the average and worst result. Compared with FCP, RTR reduces wasted computation by 83.1% and wasted transmission by 75.6%. FCP does not work well because it has to try every possible link to reach the destination before discarding packets. As a result, FCP calculates a recovery path many times and forwards packets for many hops.

Table 4.5. The wasted computation of RTR and FCP in irrecoverable test cases

	Wasted computation			
	Avg RTR	Avg FCP	Max RTR	Max FCP
AS209	1	3.2	1	9
AS701	1	2.5	1	17
AS1239	1	2.4	1	7
AS3320	1	8.4	1	31
AS3549	1	18.8	1	42
AS3561	1	7.0	1	27
AS4323	1	2.7	1	11
AS7018	1	1.9	1	10
Overall	1	5.9	1	42

Table 4.6. The wasted transmission of RTR and FCP in irrecoverable test cases

	Wasted transmission			
	Avg RTR	Avg FCP	Max RTR	Max FCP
AS209	1125.0	3366.4	7098	13364
AS701	511.7	2049.5	6072	19760
AS1239	1639.3	2049.5	18720	19760
AS3320	1071.6	3798.6	7098	20560
AS3549	929.2	8133.9	7126	30334
AS3561	790.0	7381.8	13364	32860
AS4323	716.1	2185.8	7098	12336
AS7018	676.7	1616.6	6096	18468
Overall	932.5	3822.8	18720	32860

4.5 Related Work

There have been many approaches for intra-domain routing to recover from failures. A straightforward method is to accelerate the IGP convergence via appropriately tuning the parameters related to failure detection, topology update dissemination, and routing path recomputation [65]. However, rapidly triggering the IGP convergence may cause route flapping and make the network more unstable [66]. Proactive recovery methods based on precomputed backup routes, such as [32, 36, 40], can effectively handle sporadic link failures with low overhead. However, it is hard to apply them to geographically correlated failures, because a routing path and the corresponding backup routes may fail together when geographically correlated failures occur.

The first phase of the proposed RTR looks like geographic routing in wireless networks [68, 74], but they are quite different. The only similarity is that both of them use the right-hand rule to select a node to forward packets. The first phase of RTR aims at forwarding packets around the failure area, while geographic routing in wireless networks tries to greedily forward packets to the destination. Additionally, as explained in Section 4.3.3 the forwarding disorder problem in RTR is distinct from the permanent loop problem in geographic routing. The solutions to the latter problem cannot solve the forwarding disorder problem in RTR.

There are some works related to geographically correlated failures in IP networks. Zlatokrilov *et al.* [35] proposed an approach to find paths to bypass certain failure areas in the network using distance-vector protocols. However, their method relies on the assumption that the failure area is a circle whose center and radius are known in advance. Moreover, their approach assumes that a router can query routing information from other routers. Under geographically correlated failures, however, it is a challenge to deliver query requests to other routers. Neumayer *et al.* [12, 13] analyzed the impact of geographically correlated failures on network routing, but they did not propose a recovery method. Further, they model the failure area as a circle, while we do not have any assumption on the shape of the failure area. Zheng *et al.* [63, 75] proposed an approach to identify links with abnormal performance, but they did not address how to recover the abnormal links.

Similar to RTR, the proactive recovery approach FCP [33] uses the packet header to collect failure information and calculates recovery paths on-demand. However, FCP needs to modify the mechanism of topology update dissemination in the current IP networks. Moreover, as shown in the simulation, RTR outperforms FCP in both recoverable and irrecoverable test cases. R3 [40] uses multiple backup paths to recover networks from multiple link failures. However, routers are assumed to broadcast failure information immediately after detecting a failure; whereas RTR does not make this assumption.

4.6 Conclusions

We proposed a Reactive Two-phase Rerouting (RTR) approach for intra-domain routing to quickly recover from geographically correlated failures with the shortest

recovery paths. To be more practical, we model geographically correlated failures as occurring over a continuous area of arbitrary shape and at arbitrary location. We prove that RTR is free of permanent loops. Under a single link failure, RTR guarantees to recover all failed routing paths with the shortest recovery paths. Moreover, for any failure area, the recovery paths provided by RTR are guaranteed to be the shortest. Extensive simulations based on ISP topologies show that RTR can find the shortest recovery paths for more than 98.6% of failed routing paths with reachable destinations. Compared with prior works, RTR can find more shortest recovery paths with lower computational and transmission overhead. For failed routing paths with unreachable destinations, RTR can quickly identify them and thus save network resources. Compared with prior works, RTR can save 83.1% of computation and 75.6% of transmission for irrecoverable failed routing paths.

Chapter 5

Cross-Layer Approaches for Recovery from Correlated Link Failures

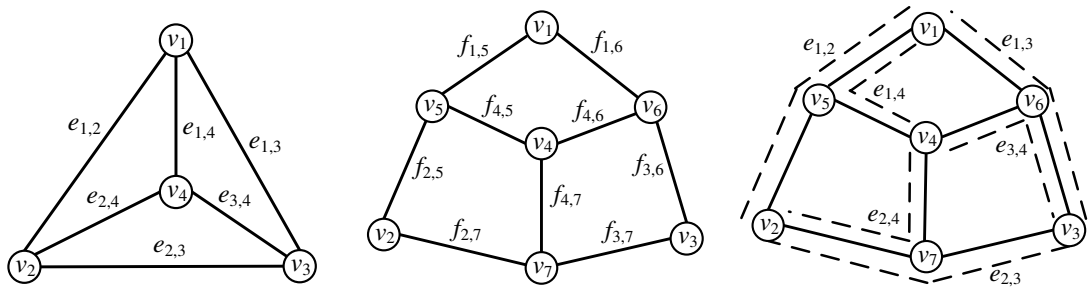
5.1 Introduction

Backup path-based protection [76, 39] is widely used by Internet Service Providers (ISPs) to recover intra-domain routing from IP link failures. In this approach, backup paths are precomputed, configured, and stored in routers. When an IP link failure is detected, traffic originally traversing the IP link is immediately switched to the backup path. Through this, the routing disruption duration can be reduced to the failure detection time which is typically less than 10 milliseconds [65].

Selecting backup paths is a critical problem in backup path-based protection. Existing approaches mainly focus on choosing reliable backup paths to protect IP links. However, they suffer from two limitations. First, the widely used failure models do not accurately reflect the correlation between IP link failures. As a result, the selected backup paths may be unreliable. Second, most prior works consider backup path selection as a connectivity problem, but ignore the traffic load and bandwidth constraint of the IP links.

Most IP backbone networks are built on the Wavelength Division Multiplexing (WDM) infrastructure [18]. In this layered structure, the IP layer topology

(logical topology) is embedded on the optical layer topology (physical topology), and each IP link (logical link) is mapped to a lightpath in the physical topology. A logical link may consist of multiple fiber links and a fiber link may be shared by multiple logical links. When a fiber link fails, all the logical links embedded on it fail simultaneously. Fig. 5.1 shows an example of the topology mapping. The logical topology in Fig. 5.1(a) is embedded on the physical topology shown in Fig. 5.1(b), where nodes v_5, v_6, and v_7 are optical layer devices and hence do not appear in the logical topology. Logical links are mapped to lightpaths as shown in Fig. 5.1(c). In the past, the optical layer had protection mechanisms to deal with fiber link failures. Such protection usually requires hardware redundancy and is quite expensive [77]. Therefore, most ISPs removed optical layer protection and use backup paths in the IP layer to restore the connectivity [18]. As a result, the model of logical link failures used for backup path selection should reflect the correlation between the logical and physical topologies.



(a) Logical topology. (b) Physical topology. (c) The mapping between the logical and fiber links.

Figure 5.1. An example of the mapping between the logical and physical topologies.

In prior works, logical link failures are considered as independent events [32, 34, 36, 38, 42] or modeled as a Shared Risk Link Group (SRLG[1]) [78, 79, 37]. However, both models have limitations. First, logical link failures are not independent because of the topology mapping. Second, sharing fiber links does not imply logical links in the same SRLG must fail simultaneously. Fig. 5.1 shows an example. If the failure of $e_{1,4}$ is caused by $f_{1,5}$, $e_{1,2}$ also fails. If it is caused by $f_{4,5}$, $e_{1,2}$ may be live. Logical links $e_{1,2}$ and $e_{1,4}$ are in the same SRLG, but they need

[1]A SRLG is a set of logical links that share the same risk such as the fiber link failure. If an IP link fails, all the IP links within the same SRLG are considered as failed.

not fail simultaneously. Their failures are correlated with a certain probability. This feature cannot be modeled by the traditional independent and SRLG models, and has not been studied in backup path selection.

Existing approaches focus on selecting reliable backup paths, but ignore the fact that a backup path may not have enough bandwidth for the rerouted traffic. Consequently, the rerouted traffic load on some logical links may exceed their usable bandwidth, and thus cause logical link overload. As Iyer *et al.* observed on a major IP backbone [80], most logical link overload is caused by the traffic rerouted due to IP link failures. In a survey in 2010, two of the largest ISPs in the world reported congestion caused by rerouted traffic in their networks [40]. Therefore, backup paths should be carefully chosen to avoid logical link overload.

We propose two cross-layer approaches built on backup paths for intra-domain routing to recover from correlated link failures. In the first approach, we develop a correlated failure probability (CFP) model based on the topology mapping and failure probability of fiber links. The CFP model quantifies the impact of IP link failure on the reliability of backup paths. With the CFP model, we propose two algorithms for selecting a backup path to protect each IP link. The first algorithm focuses on choosing the backup paths with minimum failure probability. The second algorithm further considers the bandwidth constraint and aims at minimizing the traffic disruption caused by failures. Furthermore, it controls the rerouted traffic load not to exceed the usable bandwidth of IP links to avoid interfering with the normal traffic. In the second approach, we develop a probabilistically correlated failure (PCF) model based on the topology mapping, failure probability of fiber links, and failure probability of IP links. In the PCF model, an IP link has both independent and correlated failures, which matches the discovery of the recent Internet measurements [1, 2]. With the PCF model, we propose an algorithm to choose multiple reliable backup paths to protect each IP link. When an IP link fails, its traffic is split onto multiple backup paths to ensure that the rerouted traffic load on each IP link does not exceed the usable bandwidth.

Our cross-layer approaches are different from prior works in three aspects. First, they are based on a cross-layer design, which consider the correlation between the logical and physical topologies. The proposed models can reflect the correlation between logical link failures. Second, our approaches address reliability and avoid

logical link overload. They guarantee that the rerouted traffic load does not exceed the usable bandwidth of logical links. Third, we protect each IP link with multiple backup paths to effectively reroute traffic and avoid link overload, whereas most prior works select single backup path for each logical link. We evaluate the proposed approaches using real ISP networks with both optical and IP layer topologies. Compared with existing works, the backup paths selected by our approaches are much more reliable. Moreover, the proposed approaches achieve higher recovery rate without causing logical link overload.

The rest of this chapter is organized as follows. Section 5.2 presents the preliminaries of our work. Section 5.3 presents the first approach which protects each logical link with single backup path. Section 5.4 presents the second approach which protects each logical link with multiple backup paths. Section 5.5 reviews related work, and Section 5.6 concludes the chapter.

5.2 Preliminaries

This section introduces backup path-based IP link protection and a model of IP-over-WDM networks.

5.2.1 Backup Path-Based IP Link Protection

Backup path-based IP link protection is widely used for intra-domain routing. In this approach, a router precomputes a backup path for each of its logical links. When a logical link fails, only the two routers connected by it can detect the failure. Upon detecting the failure, the router immediately switches the traffic originally sent on this logical link onto the corresponding backup path. The routing protocol has its mechanism to disseminate failure information and trigger the convergence process. Since most IP link failures are temporary and last for few seconds, rapidly triggering the convergence may cause route flapping and make networks more unstable [66]. Therefore, routers need to wait for several seconds before disseminating failure information.

The backup path is used to reroute traffic, until the routing protocol converges to a new network topology. Then, routers compute backup paths based on the

new network topology. In Fig. 5.1(a), suppose that node v_1 adopts $v_1 \rightarrow v_2 \rightarrow v_4$ as the backup path for $e_{1,4}$. When v_1 detects that $e_{1,4}$ fails, it forwards the packets towards v_4 along $v_1 \rightarrow v_2 \rightarrow v_4$. After the routing protocol converges to a new network topology, v_1 has a new shortest path towards v_4, and thus it stops using the backup path. Backup paths can be implemented with Multi-Protocol Label Switching [81] which is widely supported in the current Internet. Each backup path is configured as a Label-Switched Path (LSP) and routers can control the rerouted traffic load on a backup path.

5.2.2 Model of IP-over-WDM Networks

The IP-over-WDM network under study has a logical topology and a physical topology, which are commonly modeled as two undirected graphs [82, 83, 84]. In the physical topology $G_P = (V_P, F_P)$, V_P is a set of nodes and F_P is a set of fiber links. The fiber link from node $v_i \in V_P$ to node $v_j \in V_P$ is denoted by $f_{i,j}$. In the logical topology $G_L = (V_L, E_L)$, $V_L \subseteq V_P$ and E_L is a set of logical links. The logical link from node $v_m \in V_L$ to node $v_n \in V_L$ is denoted by $e_{m,n}$. Each logical link is mapped on the physical topology as a lightpath, i.e., a path over the fiber links. Hence, a logical link is *embedded* on fiber links, or a fiber link *carries* logical links. The topology mapping is established during network configuration, and thus is known to us. Unlike logical link states, the topology mapping is quite stable and does not change frequently. When the network administrator adjusts the topology mapping, the topology mapping information at routers can also be updated.

5.3 Protect Each IP Link With Single Backup Path

This section presents our first cross-layer approach which protects each IP link with single backup path. We first describe the CFP model in Section 5.3.1, and then introduce two algorithms for selecting backup paths in Section 5.3.2. Finally, Section 5.3.3 presents the performance evaluation. Table 5.1 summarizes the notations used in this section.

Table 5.1. Table of notations used in Section 5.3

Symbols	Meaning
	Section 5.3.1
$f_{i,j}$	fiber link from node $v_i \in V_P$ to node $v_j \in V_P$
$e_{m,n}$	logical link from node $v_m \in V_L$ to node $v_n \in V_L$
$p_{i,j}$	failure probability of $f_{i,j}$, where $p_{i,j} \in [0, 1)$
$a_{i,j}^{m,n}$	1 if $e_{m,n}$ is embedded on $f_{i,j}$, 0 otherwise
$P(e_{m,n})$	failure probability of $e_{m,n}$
$S_{m,n}^{s,t}$	set of fiber links shared by $e_{s,t}$ and $e_{m,n}$
$P(f_{i,j}\|e_{m,n})$	failure probability of $f_{i,j}$ when $e_{m,n}$ fails
$P(e_{s,t}\|e_{m,n})$	failure probability of $e_{s,t}$ when $e_{m,n}$ fails
$B_{m,n}$	backup path for logical link $e_{m,n}$
$x_{s,t}^{m,n}$	1 if $B_{m,n}$ traverses $e_{s,t}$, 0 otherwise
$F_{m,n}$	set of fiber links traversed by $B_{m,n}$
$S_B^{m,n}$	set of fiber links shared by $B_{m,n}$ and $e_{m,n}$
$P(B_{m,n}\|e_{m,n})$	failure probability of $B_{m,n}$ when $e_{m,n}$ fails
	Section 5.3.2.2
$c_{m,n}$	capacity of $e_{m,n}$
$l_{m,n}$	traffic load on $e_{m,n}$ under normal conditions
$r_{m,n}$	rerouted traffic load on $B_{m,n}$
$D_{m,n}$	traffic disruption of $e_{m,n}$

5.3.1 Correlated Failure Probability Model

We first provide some motivation and then present the details of this model. Finally, we use an example to show how the model works and why it is different from the traditional independent and SRLG models.

5.3.1.1 Motivation

Due to the topology mapping, a logical link $e_{m,n}$ may share fiber links with other logical links. When $e_{m,n}$ fails, routers cannot determine which fiber links cause the failure, and thus cannot know which logical links fail with $e_{m,n}$. Therefore, the failure of $e_{m,n}$ implies that the logical links sharing at least one fiber link with $e_{m,n}$ may also fail with a certain probability. This correlation cannot be modeled by the traditional independent and SRLG models as a *none-or-all* relation. The independent model considers that logical links only have independent failures, whereas the SRLG model considers that logical links only have correlated failures.

We develop a CFP model to deal with the correlation between logical link failures. The objective is to quantify the impact of a logical link failure on the failure probability of other logical links and backup paths. With this model, we can choose reliable backup paths to protect logical links.

5.3.1.2 The CFP Model

The CFP model is built on two kinds of information, i.e., the topology mapping and the failure probability of fiber links, both of which are already gathered by ISPs. ISPs configure their topology mapping, and thus they have this information. The failure probability of fiber links can be obtained with the Internet measurement approach [1] deployed at the optical layer, which can detect fiber link failures through SONET alarms. ISPs also maintain failure information, because they monitor the optical layer of their networks.

A key observation is that the backup path for logical link $e_{m,n}$ is used only when $e_{m,n}$ fails. Therefore, the failure probability of the backup path should be computed under the condition that $e_{m,n}$ fails. To compute the failure probability of backup paths, we first compute the failure probability of fiber links and then that of logical links.

Unlike logical links, most fiber link failures are independent [37, 84]. We assume that a fiber link $f_{i,j}$ fails independently with probability $p_{i,j} \in [0, 1)$. In practice, we may obtain $p_{i,j}$ based on previous fiber link failures. Let $a_{i,j}^{m,n}$ express the mapping between logical link $e_{m,n}$ and fiber link $f_{i,j}$.

$$a_{i,j}^{m,n} = \begin{cases} 1 & \text{if } e_{m,n} \text{ is embedded on } f_{i,j} \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

When a fiber link fails, the logical links embedded on it all fail. Let $P(e_{m,n})$ denote the failure probability of logical link $e_{m,n}$, which is computed by Eq. (5.2).

$$P(e_{m,n}) = 1 - \prod_{f_{i,j} \in F_P} (1 - a_{i,j}^{m,n} p_{i,j}) \tag{5.2}$$

Note that $P(e_{m,n})$ is the failure probability of $e_{m,n}$, but it cannot reveal the correlation between logical link failures. Since logical links are embedded on fiber links, we first investigate how a logical link failure affects the failure probability

of fiber links. Let $P(f_{i,j}|e_{m,n})$ be the failure probability of $f_{i,j}$ under the condition that $e_{m,n}$ fails. We only need to deal with the fiber link $f_{i,j}$ with failure probability $p_{i,j} > 0$. There are two cases.

- Case 1: $e_{m,n}$ is not embedded on $f_{i,j}$. It means that the failure of $e_{m,n}$ is not related with the failure of $f_{i,j}$. Therefore, $P(f_{i,j}|e_{m,n})$ is equal to $p_{i,j}$.
- Case 2: $e_{m,n}$ is embedded on $f_{i,j}$. Because $e_{m,n}$ fails, it indicates that $P(e_{m,n}) \neq 0$. Based on Bayes' theorem, $P(f_{i,j}|e_{m,n})$ is calculated as in Eq. (5.3). $P(f_{i,j})$ is the failure probability of $f_{i,j}$, which is equal to $p_{i,j}$. $P(e_{m,n}|f_{i,j})$ is the failure probability of $e_{m,n}$ under the condition that $f_{i,j}$ fails. Since $e_{m,n}$ is embedded on $f_{i,j}$, a failure of $f_{i,j}$ must result in a failure of $e_{m,n}$, i.e., $P(e_{m,n}|f_{i,j})$ is 1.

$$P(f_{i,j}|e_{m,n}) = \frac{P(e_{m,n}|f_{i,j})P(f_{i,j})}{P(e_{m,n})} = \frac{P(f_{i,j})}{P(e_{m,n})} = \frac{p_{i,j}}{P(e_{m,n})} \tag{5.3}$$

In summary, the failure probability $P(f_{i,j}|e_{m,n})$ is shown in Eq. (5.4).

$$P(f_{i,j}|e_{m,n}) = \begin{cases} p_{i,j} & a_{i,j}^{m,n} = 0 \\ \frac{p_{i,j}}{P(e_{m,n})} & a_{i,j}^{m,n} = 1 \end{cases} \tag{5.4}$$

Next, we calculate the failure probability of logical link $e_{s,t}$ under the condition that $e_{m,n}$ fails. Let $P(e_{s,t}|e_{m,n})$ denote this probability. Note that we cannot simply replace $p_{i,j}$ in Eq. (5.2) with $P(f_{i,j}|e_{m,n})$ to obtain $P(e_{s,t}|e_{m,n})$. In statistical theory, Eq. (5.2) is correct only when probability $p_{i,j}$ is independent to each other. However, for two fiber links $f_{i,j}$ and $f_{g,h}$, their failure probability $P(f_{i,j}|e_{m,n})$ and $P(f_{g,h}|e_{m,n})$ may not be independent. Fig. 5.1(c) shows one example. Suppose $e_{1,4}$ fails, which indicates that at least one of $f_{1,5}$ and $f_{4,5}$ fails. If $f_{1,5}$ fails, $f_{4,5}$ may be live or fail. If $f_{1,5}$ is live, $f_{4,5}$ must have failed in order for $e_{1,4}$ to fail. When $e_{1,4}$ fails, the status of $f_{1,5}$ is related with the status of $f_{4,5}$. Therefore, $P(f_{1,5}|e_{1,4})$ is not independent to $P(f_{4,5}|e_{1,4})$.

The correlation between $e_{s,t}$ and $e_{m,n}$ is because they share fiber links. To calculate $P(e_{s,t}|e_{m,n})$, we first identify the fiber links shared by $e_{s,t}$ and $e_{m,n}$. Let $S_{m,n}^{s,t}$ be the set of fiber links shared by $e_{s,t}$ and $e_{m,n}$, which is defined by Eq. (5.5).

$$S_{m,n}^{s,t} = \{f_{i,j} | a_{i,j}^{s,t} a_{i,j}^{m,n} = 1, \forall f_{i,j} \in F_P\} \tag{5.5}$$

This set of fiber links *fails* if *any* fiber link in the set fails, and it is *live* when every fiber link in the set is live. Let $P(S_{m,n}^{s,t})$ be the probability that set $S_{m,n}^{s,t}$ fails. Since the fiber links in $S_{m,n}^{s,t}$ fail independently, we can calculate $P(S_{m,n}^{s,t})$ with Eq. (5.6).

$$P(S_{m,n}^{s,t}) = \begin{cases} 1 - \prod_{f_{i,j} \in S_{m,n}^{s,t}} (1 - p_{i,j}) & \text{if } S_{m,n}^{s,t} \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

According to the definition of $S_{m,n}^{s,t}$, $e_{m,n}$ is embedded on every fiber link in $S_{m,n}^{s,t}$. Therefore, we can use the method similar to Eq. (5.4) to compute the failure probability of $S_{m,n}^{s,t}$ under the condition that $e_{m,n}$ fails, which is shown in Eq. (5.7).

$$P(S_{m,n}^{s,t} | e_{m,n}) = \frac{P(S_{m,n}^{s,t})}{P(e_{m,n})} \tag{5.7}$$

The fiber links carrying $e_{s,t}$ can be divided into two sets. The first set $S_{m,n}^{s,t}$ contains the fiber links shared by $e_{m,n}$ and $e_{s,t}$, and thus these fiber links are related with the failure of $e_{m,n}$. The second set consists of the fiber links that carry $e_{s,t}$ but do not carry $e_{m,n}$. Any fiber link $f_{i,j}$ in the second set is not related with the failure of $e_{m,n}$. Hence, $P(f_{i,j}|e_{m,n}) = p_{i,j}$ according to Eq. (5.4) and $P(f_{i,j}|e_{m,n})$ is independent to $P(S_{m,n}^{s,t}|e_{m,n})$. In summary, $P(e_{s,t}|e_{m,n})$ can be calculated with Eq. (5.8).

$$P(e_{s,t}|e_{m,n}) = 1 - (1 - P(S_{m,n}^{s,t}|e_{m,n})) \times \prod_{f_{i,j} \in F_P - S_{m,n}^{s,t}} (1 - a_{i,j}^{s,t} p_{i,j}) \tag{5.8}$$

Finally, we calculate the failure probability of the backup path $B_{m,n}$ for $e_{m,n}$ under the condition that $e_{m,n}$ fails. This probability is denoted by $P(B_{m,n}|e_{m,n})$. We define variable $x_{s,t}^{m,n} \in \{0, 1\}$ as follows.

$$x_{s,t}^{m,n} = \begin{cases} 1 & \text{if } B_{m,n} \text{ traverses logical link } e_{s,t} \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

In the independent and SRLG models, $P(B_{m,n}|e_{m,n})$ is computed based on the failure probability of logical links as shown in Eq. (5.10).

$$P(B_{m,n}|e_{m,n}) = 1 - \prod_{e_{s,t} \in E_L} (1 - x_{s,t}^{m,n} P(e_{s,t}|e_{m,n})) \tag{5.10}$$

This equation enumerates logical links and count the ones traversed by $B_{m,n}$. However, it may count a fiber link multiple times if this fiber link is shared by multiple logical links on $B_{m,n}$. As shown in Fig. 5.1, path $v_1 \rightarrow v_3 \rightarrow v_4$ traverses fiber link $f_{3,6}$ twice, and thus Eq. (5.10) counts $f_{3,6}$ twice. Actually, we should count each fiber link only once, because fiber link failures have the same effect on disconnecting $B_{m,n}$. A failure of any underlying fiber link leads to a failure of $B_{m,n}$. Therefore, we compute $P(B_{m,n}|e_{m,n})$ based on fiber links rather than logical links. Similar to the calculation of $P(e_{s,t}|e_{m,n})$, we first determine the fiber links shared by $B_{m,n}$ and $e_{m,n}$. Let $S_B^{m,n}$ be the set of these fiber links, which is defined in Eq. (5.11).

$$S_B^{m,n} = \{f_{i,j} | x_{s,t}^{m,n} a_{i,j}^{s,t} a_{i,j}^{m,n} = 1, \forall e_{s,t} \in E_L, \forall f_{i,j} \in F_P\} \tag{5.11}$$

Like set $S_{m,n}^{s,t}$, set $S_B^{m,n}$ *fails* if any fiber link in it fails, and it is live when every fiber link in it is live. Then, we calculate the failure probability $P(S_B^{m,n})$ and the conditional failure probability $P(S_B^{m,n}|e_{m,n})$ similar to $P(S_{m,n}^{s,t})$ and $P(S_{m,n}^{s,t}|e_{m,n})$. The expressions are similar to Eq. (5.6) and Eq. (5.7) (by replacing $S_{m,n}^{s,t}$ with $S_B^{m,n}$), and hence we omit them.

Finally, we compute $P(B_{m,n}|e_{m,n})$ with the method similar to the calculation of $P(e_{s,t}|e_{m,n})$ in Eq. (5.8). To simplify the expression, we introduce notation $F_{m,n}$ to be the set of fiber links traversed by $B_{m,n}$, which is shown in Eq. (5.12).

$$F_{m,n} = \{f_{i,j} | x_{s,t}^{m,n} a_{i,j}^{s,t} = 1, \forall e_{s,t} \in E_L, \forall f_{i,j} \in F_P\} \tag{5.12}$$

Note that $S_B^{m,n}$ is a subset of $F_{m,n}$. The fiber links in $F_{m,n} - S_B^{m,n}$ are not related with the failure of $e_{m,n}$, because they do not carry $e_{m,n}$. According to Eq. (5.4),

we have $P(f_{i,j}|e_{m,n}) = p_{i,j}$ for each of these fiber links. In summary, $P(B_{m,n}|e_{m,n})$ is computed by Eq. (5.13), which has two features. First, each fiber link carrying $B_{m,n}$ is counted once. Second, the fiber links related with the failure of $e_{m,n}$ are separated from the fiber links independent to the failure of $e_{m,n}$. In Section 5.3.2.1, we propose an algorithm to choose $B_{m,n}$ with minimum $P(B_{m,n}|e_{m,n})$.

$$P(B_{m,n}|e_{m,n}) = 1 - (1 - P(S_B^{m,n}|e_{m,n})) \times \prod_{f_{i,j} \in F_{m,n} - S_B^{m,n}} (1 - p_{i,j}) \tag{5.13}$$

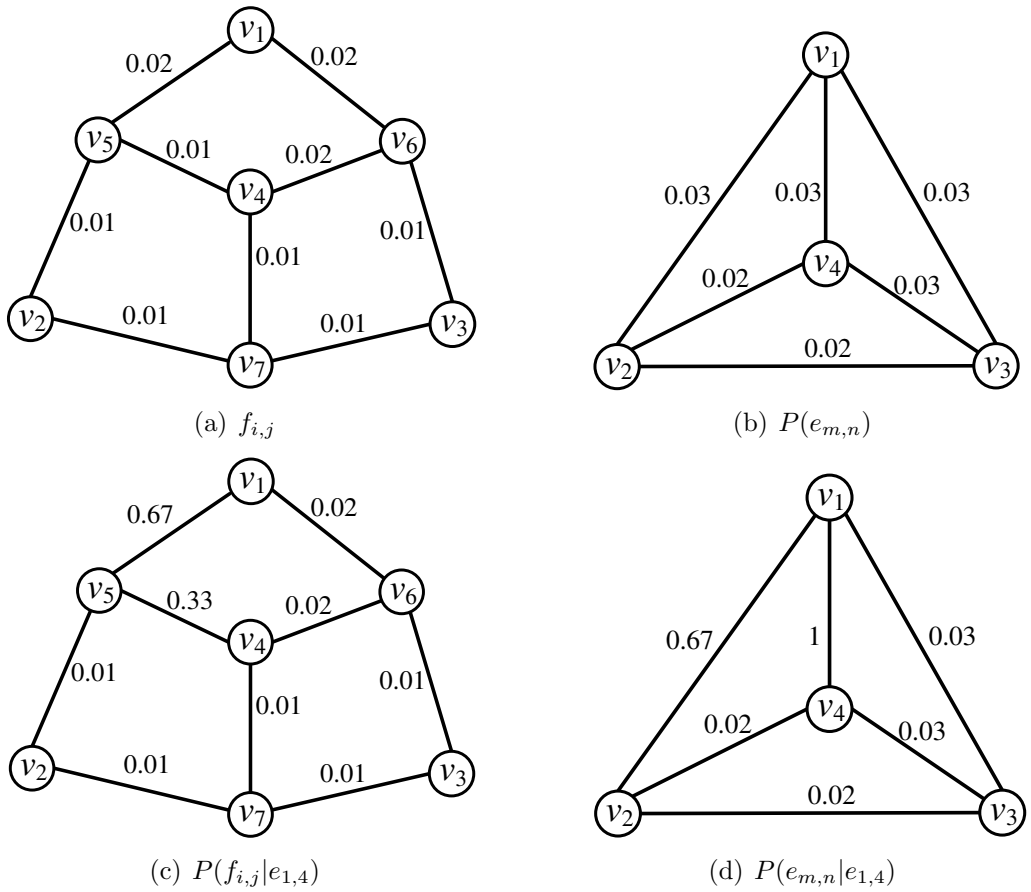
5.3.1.3 An Example

We use an example to show how the CFP model works and its difference from the traditional independent and SRLG models. The physical and logical topologies are shown in Fig. 5.1. The failure probability of fiber links is shown in Fig. 5.2(a). Suppose v_1 needs to choose a backup path for $e_{1,4}$. First, Eq. (5.2) calculates the failure probability $P(e_{m,n})$ for each logical link $e_{m,n}$. The result is shown in Fig. 5.2(b). For example, $P(e_{1,4}) = 1 - (1 - 0.02)(1 - 0.01) = 0.03$. Then, we use Eq. (5.4) to calculate the failure probability $P(f_{i,j}|e_{1,4})$. The result is shown in Fig. 5.2(c). Since $e_{1,4}$ is embedded on $f_{1,5}$ and $f_{4,5}$, $P(f_{1,5}|e_{1,4})$ is different from $p_{1,5}$ and $P(f_{4,5}|e_{1,4})$ is different from $p_{4,5}$. For example, $P(f_{1,5}|e_{1,4}) = \frac{p_{1,5}}{P(e_{1,4})} = \frac{0.02}{0.03} = 0.67$. Next, we use Eq. (5.5)–Eq. (5.8) to compute $P(e_{m,n}|e_{1,4})$ and show the result in Fig. 5.2(d). For $e_{1,2}$, set $S_{1,2}^{1,4}$ contains $f_{1,5}$, because $e_{1,4}$ and $e_{1,2}$ only share $f_{1,5}$. Hence, the failure probability $P(S_{1,2}^{1,4}) = 1 - (1 - p_{1,5}) = 0.02$ and $P(S_{1,2}^{1,4}|e_{1,4}) = \frac{P(S_{1,2}^{1,4})}{P(e_{1,4})} = \frac{0.02}{0.03} = \frac{2}{3}$. Finally, $P(e_{1,2}|e_{1,4}) = 1 - (1 - P(S_{1,2}^{1,4}|e_{1,4}))(1 - p_{2,5}) = 1 - (1 - \frac{2}{3})(1 - 0.01) = 0.67$.

Table 5.2. The failure probability $P(e_{m,n}|e_{1,4})$ computed by the three models

Model name	$e_{1,2}$	$e_{1,3}$	$e_{1,4}$	$e_{2,3}$	$e_{2,4}$	$e_{3,4}$
Independent	0.03	0.03	1	0.02	0.02	0.03
SRLG	1	0.03	1	0.02	0.02	0.03
CFP	0.67	0.03	1	0.02	0.02	0.03

Table 5.2 summarizes the failure probability $P(e_{m,n}|e_{1,4})$ computed by the two existing models and our model. The models lead to different result for $e_{1,2}$, because only $e_{1,2}$ shares a fiber link with $e_{1,4}$. Due to the omission of topology correlation,



(a) $f_{i,j}$ (b) $P(e_{m,n})$

(c) $P(f_{i,j}|e_{1,4})$ (d) $P(e_{m,n}|e_{1,4})$

Figure 5.2. An example of calculating failure probability with the CFP model.

the independent model considers that logical link failures are independent, and hence it underestimates $P(e_{1,2}|e_{1,4})$. The SRLG model overestimates $P(e_{1,2}|e_{1,4})$, because it considers that the logical links sharing fiber links with $e_{1,4}$ fail simultaneously.

Table 5.3. The failure probability $P(B_{1,4}|e_{1,4})$ computed by the three models

Backup path	**Independent**	**SRLG**	**CFP**
$v_1 \rightarrow v_2 \rightarrow v_4$	0.05	1	0.68
$v_1 \rightarrow v_3 \rightarrow v_4$	0.06	0.06	0.05
$v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4$	0.08	1	0.69
$v_1 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4$	0.07	0.07	0.06

Finally, we compute the failure probability $P(B_{1,4}|e_{1,4})$. The backup path $B_{1,4}$ can be chosen from four loop-free paths as shown in the first column of Table 5.3.

The independent and SRLG models compute $P(B_{1,4}|e_{1,4})$ with Eq. (5.10) based on the results in Table 5.2. The result is shown in the second and third columns. We use $v_1 \rightarrow v_2 \rightarrow v_4$ as an example to show how our model computes $P(B_{1,4}|e_{1,4})$. The set $F_{1,4}$ consists of four fiber links $f_{1,5}$, $f_{2,5}$, $f_{2,7}$, and $f_{4,7}$. Since this path shares $f_{1,5}$ with $e_{1,4}$, the set $S^B_{1,4}$ contains $f_{1,5}$. Hence, $P(S^B_{1,4}) = p_{1,5} = 0.02$ and $P(S^B_{1,4}|e_{1,4}) = \frac{0.02}{0.03} = \frac{2}{3}$. Finally, Eq. (5.13) calculates $P(B_{1,4}|e_{1,4}) = 1 - (1 - P(S^B_{1,4}|e_{1,4}))(1 - p_{2,5})(1 - p_{2,7})(1 - p_{4,7}) = 1 - (1 - \frac{2}{3})(1 - 0.01)^3 \approx 0.68$.

The independent and SRLG models have two limitations in calculation of $P(B_{1,4}|e_{1,4})$. First, the two models do not accurately reflect the correlation between logical link failures. As a result, the independent model may underestimate $P(B_{1,4}|e_{1,4})$ and the SRLG model may overestimate it as shown by the first and third rows of Table 5.3. Second, they may count a fiber link multiple times. For example, the two models count $f_{3,6}$ twice for $v_1 \rightarrow v_3 \rightarrow v_4$, and count $f_{2,7}$ twice for $v_1 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4$. However, each fiber link carrying the backup path should be counted once, because they have the same effect on disconnecting the backup path.

5.3.2 Backup Path Selection

Based on the CFP model, we propose two algorithms to select backup paths. The first algorithm focuses on reliability and aims at choosing the backup paths with minimum failure probability. The second algorithm further considers the bandwidth constraint and aims at choosing backup paths to minimize the traffic disruption caused by failures. Furthermore, it controls the rerouted traffic load to prevent causing logical link overload.

5.3.2.1 Reliability Oriented Backup Paths

Reliability oriented backup paths aim at maximizing the reliability of backup paths without considering the traffic load of rerouted traffic. Most prior works focus on this objective. In this problem, backup paths do not compete for bandwidth, and thus we can choose them one by one. For each logical link $e_{m,n}$, the objective is to choose the backup path $B_{m,n}$ with minimum failure probability $P(B_{m,n}|e_{m,n})$. Since $B_{m,n}$ is used only when $e_{m,n}$ fails, the objective is equal to minimizing the

probability that $e_{m,n}$ and $B_{m,n}$ fail simultaneously.

This problem can be formally defined as an optimization problem as shown in Eq. (5.14)–Eq. (5.16). The objective function Eq. (5.14) means that we aim at selecting the backup path $B_{m,n}$ with minimum $P(B_{m,n}|e_{m,n})$ to protect logical link $e_{m,n}$. $x_{i,j}^{m,n}$ is defined in Eq. (5.9) to show if $B_{m,n}$ traverses $e_{i,j}$. The connectivity constraint in Eq. (5.15) requires that the selected logical links form a path from v_m to v_n.

$$\text{minimize} \quad P(B_{m,n}|e_{m,n}) \tag{5.14}$$

subject to

$$\sum_{j:e_{i,j}\in E_L} x_{i,j}^{m,n} - \sum_{j:e_{j,i}\in E_L} x_{j,i}^{m,n} = \begin{cases} 1 & i = m \\ -1 & i = n \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

$$\forall e_{i,j} \in E_L, \quad x_{i,j}^{m,n} \in \{0, 1\} \tag{5.16}$$

The above formulation is a 0-1 nonlinear programming problem, which is NP-hard in general. Since routers calculate backup paths, the algorithm should be simple and have low computational complexity. Complex algorithms consume much computational resources of the routers, and thus affect normal packet forwarding. Therefore, we propose a greedy algorithm SelectBP as shown in Algorithm 5 to efficiently solve it. The basic idea is similar to Dijkstra's algorithm for calculating the shortest path. Starting from node v_m, the algorithm gradually adds logical links to expand the backup path, until finding $B_{m,n}$ with minimum $P(B_{m,n}|e_{m,n})$. By storing $B_{m,i}$ for v_i, constructing $B_{m,j}$ in line 16 can be achieved in $O(1)$ time. Therefore, the computational complexity of Algorithm 5 is $O((|E_L|+|V_L|)\log|V_L|)$.

5.3.2.2 Backup Paths with Bandwidth Constraint

Without considering the rerouted traffic load, reliability oriented backup paths may overload logical links, and thus interfere with normal traffic. To address

Algorithm 5 SelectBP

Procedure:

```
1: G_L ← G_L − e_{m,n}
2: for each node v_i ∈ V_L do
3:    w(v_i) ← ∞    ▷ weight of node v_i, which is the failure probability
      P(B_{m,i}|e_{m,n}) of B_{m,i} from v_m to v_i.
4:    visit(v_i) ← 0    ▷ flag to denote if v_i has been visited
5:    prev(v_i) ← NULL    ▷ previous node of v_i on B_{m,i}
6: end for
7: w(v_m) ← 0
8: Initialize a priority queue Q, and Q ← V_L
9: while Q ≠ ∅ do
10:    v_i ←ExtractMin(Q)    ▷ extract the node v_i with minimum weight w(v_i)
       from Q
11:    if w(v_i) = ∞ then
12:       break
13:    end if
14:    visit(v_i) ← 1
15:    for each neighbor v_j of v_i with visit(v_j) = 0 do
16:       B_{m,j} ← the path from v_m to v_j constructed from prev
17:       if w(v_j) > P(B_{m,j}|e_{m,n}) then
18:          w(v_j) ← P(B_{m,j}|e_{m,n})
19:          prev(v_j) ← v_i
20:       end if
21:    end for
22: end while
23: Construct B_{m,n} from prev
```

this problem, we choose backup paths considering the bandwidth constraint. The basic idea is to control the rerouted traffic load and only use the logical links with usable bandwidth. Let $c_{m,n}$ denote the capacity of $e_{m,n}$. Under normal conditions, the traffic load on $e_{m,n}$ is $l_{m,n}$ which satisfies $l_{m,n} < c_{m,n}$. Network administrators configure the link cost to achieve traffic engineering goals, and hence traffic load $l_{m,n}$ is known. The bandwidth of $e_{m,n}$ that can be used for backup paths is $c_{m,n} - l_{m,n}$. Logical link $e_{m,n}$ is *overloaded* if the rerouted traffic load on it exceeds $c_{m,n} - l_{m,n}$.

For a large traffic load $l_{m,n}$, there may not be any backup path which has enough bandwidth to support it. A simple solution is to leave $e_{m,n}$ unprotected to avoid logical link overload. However, all traffic on $e_{m,n}$ is disrupted when $e_{m,n}$ fails.

Therefore, we adopt another method, i.e., controlling the rerouted traffic load to protect a part of the traffic. In addition to choosing $B_{m,n}$, we determine the traffic load $r_{m,n}$ for $B_{m,n}$ satisfying $r_{m,n} \leq l_{m,n}$. When $e_{m,n}$ fails, node v_m switches the traffic originally on $e_{m,n}$ onto $B_{m,n}$ and controls the rerouted traffic load to be $r_{m,n}$. The unprotected traffic load is $l_{m,n} - r_{m,n}$, which is disrupted when $e_{m,n}$ fails.

Our objective is to minimize the traffic disruption in the entire network. Based on the CFP model, we define the traffic disruption of $e_{m,n}$ in Eq. (5.17). When $e_{m,n}$ fails, the unprotected traffic is disrupted. If $B_{m,n}$ also fails, the traffic rerouted on it is disrupted. Therefore, $D_{m,n}$ is the mathematical expectation of disrupted traffic load of $e_{m,n}$.

$$D_{m,n} = P(e_{m,n})(r_{m,n}P(B_{m,n}|e_{m,n}) + l_{m,n} - r_{m,n}) \tag{5.17}$$

The traffic disruption of the entire network is then defined in Eq. (5.18), which is the mathematical expectation of traffic disruption in the entire network.

$$D = \sum_{e_{m,n} \in E_L} D_{m,n} \tag{5.18}$$

Our problem can be formally defined as an optimization problem as shown in Eq. (5.19)–Eq. (5.22). For each logical link $e_{m,n}$, the connectivity constraint in Eq. (5.20) requires that the logical links selected by $B_{m,n}$ form a path from v_m to v_n. The constraint in Eq. (5.21) requires that the rerouted traffic load on each logical link does not exceed its usable bandwidth. Eq. (5.22) specifies that the rerouted traffic load $r_{m,n}$ does not exceed $l_{m,n}$. Ideally, all the traffic load of $e_{m,n}$ should be rerouted. In some cases, the network does not have enough bandwidth, and hence the overall rerouted traffic load of $e_{m,n}$ may be lower than $l_{m,n}$.

$$\text{minimize} \quad D \tag{5.19}$$

subject to

$$\sum_{j:e_{i,j}\in E_L} x_{i,j}^{m,n} - \sum_{j:e_{j,i}\in E_L} x_{j,i}^{m,n} = \begin{cases} 1 & i = m \\ -1 & i = n \\ 0 & \text{otherwise} \end{cases} \tag{5.20}$$

$$\sum_{e_{m,n}\in E_L} x_{i,j}^{m,n} r_{m,n} + l_{i,j} \leq c_{i,j} \tag{5.21}$$

$$\forall e_{m,n} \in E_L, \quad \forall e_{i,j} \in E_L, \quad x_{i,j}^{m,n} \in \{0,1\}, \quad 0 \leq r_{m,n} \leq l_{m,n} \tag{5.22}$$

The above formulation is a mixed integer nonlinear programming problem which is NP-hard. We propose a multi-round algorithm SelectBP-BC as shown in Algorithm 6 to efficiently solve it. The basic idea is to choose backup paths one by one. Intuitively, a logical link $e_{m,n}$ with large failure probability $P(e_{m,n})$ and large traffic load $l_{m,n}$ should be protected most, because it is easy to fail and its failure disrupts a large amount of traffic. We use $P(e_{m,n})l_{m,n}$ as the weight of $e_{m,n}$ and deal with logical links in the decreasing order of their weights. The algorithm first sort logical links in decreasing order of their weights (line 1). In each round, the algorithm picks out the unprotected logical link $e_{m,n}$ with the largest weight. It removes $e_{m,n}$ (line 9), and then uses Algorithm 5 to compute $B_{m,n}$ with minimum failure probability (line 10). Next, the algorithm determines the rerouted traffic load $r_{m,n}$ (lines 11–16), which is the smaller one between $l_{m,n}$ and the usable bandwidth of $B_{m,n}$. Finally, it updates the usable bandwidth of logical links and removes the logical links without usable bandwidth (lines 17–22). The computational complexity is $O(|E_L|(|E_L| + |V_L|)\log|V_L|)$, because the algorithm invokes Algorithm 5 for each of the $|E_L|$ logical links.

5.3.3 Performance Evaluation

We evaluate the performance of the proposed approach and compare it with other backup path-based approaches.

Algorithm 6 SelectBP-BC

Procedure:

1: Sort logical links in the decreasing order of $P(e_{m,n})l_{m,n}$
2: **for** each logical link $e_{i,j} \in$ **do**
3: $u_{i,j} \leftarrow c_{i,j} - l_{i,j}$ ▷ the usable bandwidth of $e_{i,j}$
4: **if** $u_{i,j} = 0$ **then**
5: $G_L \leftarrow G_L - e_{i,j}$ ▷ remove $e_{i,j}$ from G_L if it does not have usable bandwidth
6: **end if**
7: **end for**
8: **for** each logical link $e_{m,n} \in E_L$ **do**
9: $G_{temp} \leftarrow G_L - e_{m,n}$
10: $B_{m,n} \leftarrow$ SelectBP for $e_{m,n}$ on G_{temp}
11: $r_{m,n} \leftarrow l_{m,n}$
12: **for** each logical link $e_{i,j}$ on $B_{m,n}$ **do**
13: **if** $u_{i,j} < r_{m,n}$ **then**
14: $r_{m,n} \leftarrow u_{i,j}$
15: **end if**
16: **end for**
17: **for** each logical link $e_{i,j}$ on $B_{m,n}$ **do**
18: $u_{i,j} \leftarrow u_{i,j} - r_{m,n}$
19: **if** $u_{i,j} = 0$ **then**
20: $G_L \leftarrow G_L - e_{i,j}$
21: **end if**
22: **end for**
23: **end for**

5.3.3.1 Simulation Setup

Network topology: The simulation is based on four real ISP networks with both physical and logical topologies, i.e., ChinaNet[2] [3], Level3[4], Qwest[5], and XO[6]. We use PoP-level logical topology, where nodes correspond to cities. Table 5.4 summarizes the physical and logical topologies in the four networks. Due to lack of lightpath information, we build the topology mapping with the method used in [82] to minimize the number of fiber links shared by logical links.

[2] `http://www.chinatelecomusa.com/content_images/NationalFiber_Full.jpg`
[3] `http://www.chinatelecomusa.com/content_images/ChinaNet_Full.jpg`
[4] `http://www.level3.com/en/About-Us/~/media/Assets/maps/level_3_network_map.ashx`
[5] `http://www.qwest.com/largebusiness/enterprisesolutions/networkMaps/`
[6] `http://www.xo.com/about/network/Pages/maps.aspx`

Table 5.4. Real ISP networks used for evaluation

Network	Physical topology		Logical topology	
	# Nodes	# Fiber links	# Nodes	# Logical links
ChinaNet	85	141	39	61
Level3	209	230	63	285
Qwest	152	181	49	77
XO	61	71	40	65

IP layer configuration: The logical link capacity of Qwest and XO networks is set to that provided in their logical topologies. For ChinaNet and Level3, we use the method in [85] to assign the logical link capacity. This method assumes that the high degree nodes are Level-1 PoP. The logical links between Level-1 PoP have high capacity (10Gb/s) and other logical links have low capacity (2.5Gb/s), which matches the recent ISP case studies [86]. Similar to [85, 40], we use the gravity model to generate synthetic traffic demands. This method assumes that the incoming traffic at a PoP is proportional to the combined capacity of its outgoing links. The average logical link utilization of generated traffic is from 5% to 40% in increments of 5%. We use 40% as the upper bound because ISPs usually upgrade their infrastructure when the logical link utilization exceeds 40% [86, 85]. Therefore, it is important to investigate the performance when the logical link utilization is up to 40%. Finally, we use the method in [87] to optimize the link cost based on the link capacity and traffic demands. Each logical topology adopts the shortest path routing calculated based on the optimized link cost.

Failure scenarios: Generally, the failure probability of the fiber link is quite low. In the simulation, it is a random number between 1% and 0.01% with the precision of 0.01%. We focus on the scenario of single fiber link failure, which may cause multiple logical link failures. In the simulation, we use 100 failure probability settings. For each setting, the simulation is run 1,000 times and each time one fiber link is selected to fail based on the failure probability. The logical links embedded on the failed fiber link all fail. A fiber link failure is counted as one test case, and thus each simulation has 100,000 test cases.

Algorithms: We compare our algorithms with Not-via [88] and PSRLG-based Diverse Routing (DR) with disjointness constraint [37]. Not-via is an IP fast-rerouting (IPFRR) technique widely deployed in the Internet. DR uses a parameter

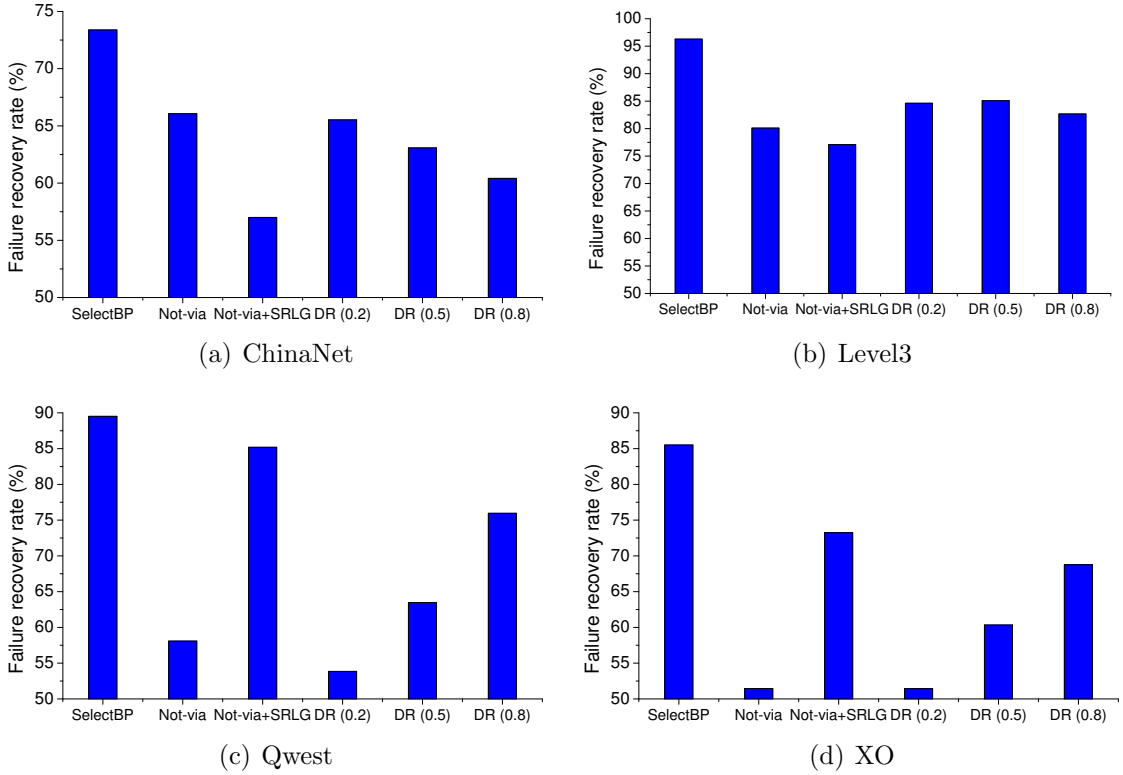
p to specify the failure probability of logical links when the underlying fiber link fails. We set p to three typical values 0.2, 0.5, and 0.8. In summary, we compare our algorithms with five algorithms as follows.

- *Not-via*: Not-via built on the independent model.
- *Not-via+SRLG*: Not-via built on the SRLG model.
- *DR (0.2)*: DR with its parameter p of 0.2.
- *DR (0.5)*: DR with its parameter p of 0.5.
- *DR (0.8)*: DR with its parameter p of 0.8.

5.3.3.2 Reliability of Backup Paths

We first investigate the reliability of the backup paths. For our cross-layer approach, we use the algorithm SelectBP proposed in Section 5.3.2.1 to choose backup paths. In a test case, if a failed logical link has a live backup path, this logical link failure can be recovered. The *failure recovery rate* is the percentage of the recovered logical link failures, and it is used as the performance metric.

The average failure recovery rate across 100,000 test cases is shown in Fig. 5.3, and the overall failure recovery rate for the four networks is shown in Fig. 5.4. We highlight three features. First, our algorithm SelectBP outperforms the other five algorithms in all four networks. It shows that our cross-layer design and backup path selection algorithm are effective for finding reliable backup paths. Not-via ignores the correlation between logical link failures, and thus backup paths may traverse some failed logical links. Not-via+SRLG may remove some useful logical links and even disconnect the topology. Consequently, some logical links may not have backup paths. Second, unlike SelectBP, the performance of the other five algorithms is not consistent across the four networks. For example, Not-via is the second best in ChinaNet, but it is the worst in XO. Similarly, Not-via+SRLG is the worse in ChinaNet and Level3, while it is the second best in Qwest and XO. Third, the parameter p in DR strongly affects the performance of DR. It is difficult to choose an appropriate p to achieve good performance in all networks.



(a) ChinaNet (b) Level3

(c) Qwest (d) XO

Figure 5.3. The average failure recovery rate.

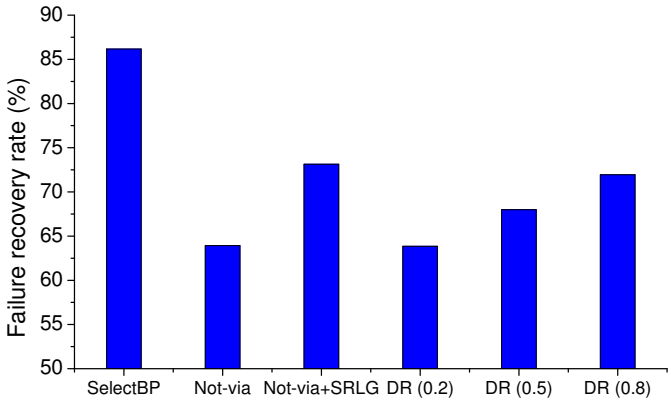


Figure 5.4. The overall failure recovery rate for the four networks.

A fiber link failure may cause multiple logical link failures. In some cases, all these failed logical links are recovered; while in some other cases, only some failed logical links are recovered. We count the cases that all failed logical links are recovered. The percentage of these test cases is summarized in Table 5.5. Similar to the failure recovery rate, SelectBP outperforms the other fiver algorithms in all

four networks.

Table 5.5. Percentage of the test cases that failed logical links are all recovered

Algorithm	ChinaNet	Level3	Qwest	XO
SelectBP	56.3	92.9	85.2	83.8
Not-via	47.8	62.6	37.0	29.6
Not-via+SRLG	36.8	57.7	75.3	54.0
DR (0.2)	48.8	69.8	35.3	36.6
DR (0.5)	44.4	68.9	40.9	40.5
DR (0.8)	40.8	64.7	55.1	47.7

5.3.3.3 Traffic Recovery Rate and Logical Link Overload

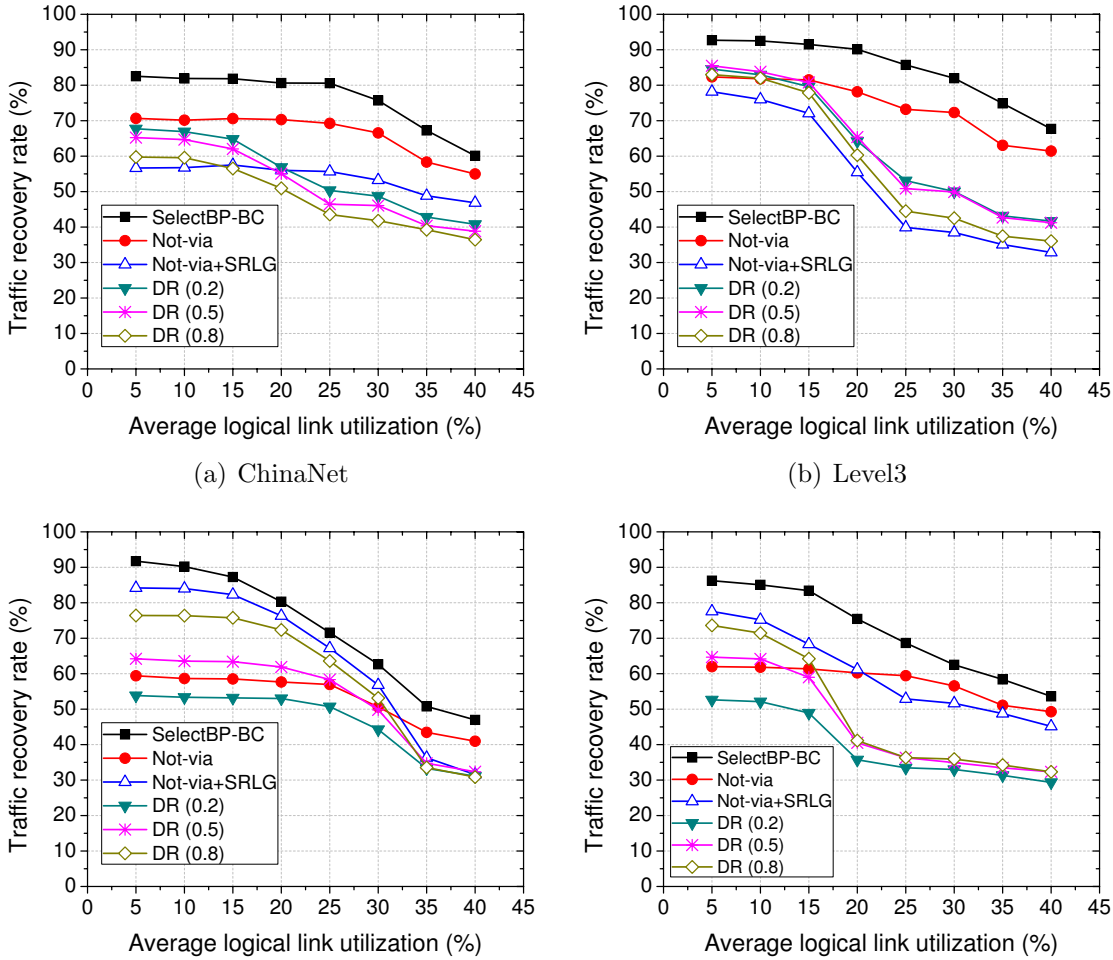
Next, we consider the bandwidth constraint and evaluate the performance of the algorithm SelectBP-BC proposed in Section 5.3.2.2. We define the following two metrics for measuring the benefit and negative impact.

- *Traffic recovery rate*: Suppose logical link $e_{m,n}$ fails. If its backup path $B_{m,n}$ does not contain any failed or overloaded logical link, the traffic of $e_{m,n}$ rerouted by $B_{m,n}$ is recovered. The recovered traffic load in SelectBP-BC is $r_{m,n}$, because it controls the rerouted traffic load on $B_{m,n}$. The other five algorithms do not control the rerouted traffic load, and thus the recovered traffic load is $l_{m,n}$. Suppose the overall traffic load of all failed logical links is T. We measure the recovered traffic load T_r achieved by each algorithm. The traffic recovery rate is defined as $\frac{T_r}{T}$. The optimal value (100%) means that no traffic is disrupted by failures.

- *Overload rate*: In a test case, we count the logical links traversed by the rerouted traffic and denote this number as L. We also count the overloaded ones among them. A logical link is overloaded if its capacity is smaller than the traffic load on it, including its own traffic and the rerouted traffic. Suppose there are L_o overloaded logical links. The overload rate is defined as $\frac{L_o}{L}$, and it quantifies the negative impact caused by the rerouted traffic.

The average traffic recovery rate is shown in Fig. 5.5, and the overall result for the four networks is shown in Fig. 5.6. We highlight three important aspects of

the simulation results. First, SelectBP-BC outperforms the other five algorithms under different logical link utilizations in each network. Second, the performance of the other five algorithms is not consistent across the four networks, which is similar to the failure recovery rate in Fig. 5.3. Our approach is better than the other five in adapting to different networks and logical link utilizations. Third, the traffic recovery rate decreases as the logical link utilization increases due to lack of usable bandwidth. For example, if the logical link utilization is 25%, the usable bandwidth is 3 times the traffic load. However, when it increases to 40%, the usable bandwidth decreases to 1.5 times the traffic load. A small increase in the traffic load makes rerouting much more difficult.



(a) ChinaNet (b) Level3

(c) Qwest (d) XO

Figure 5.5. The average traffic recovery rate under different logical link utilizations.

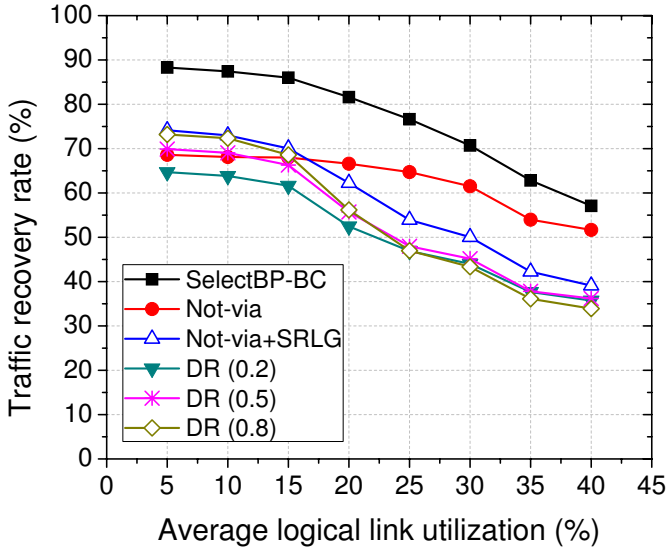
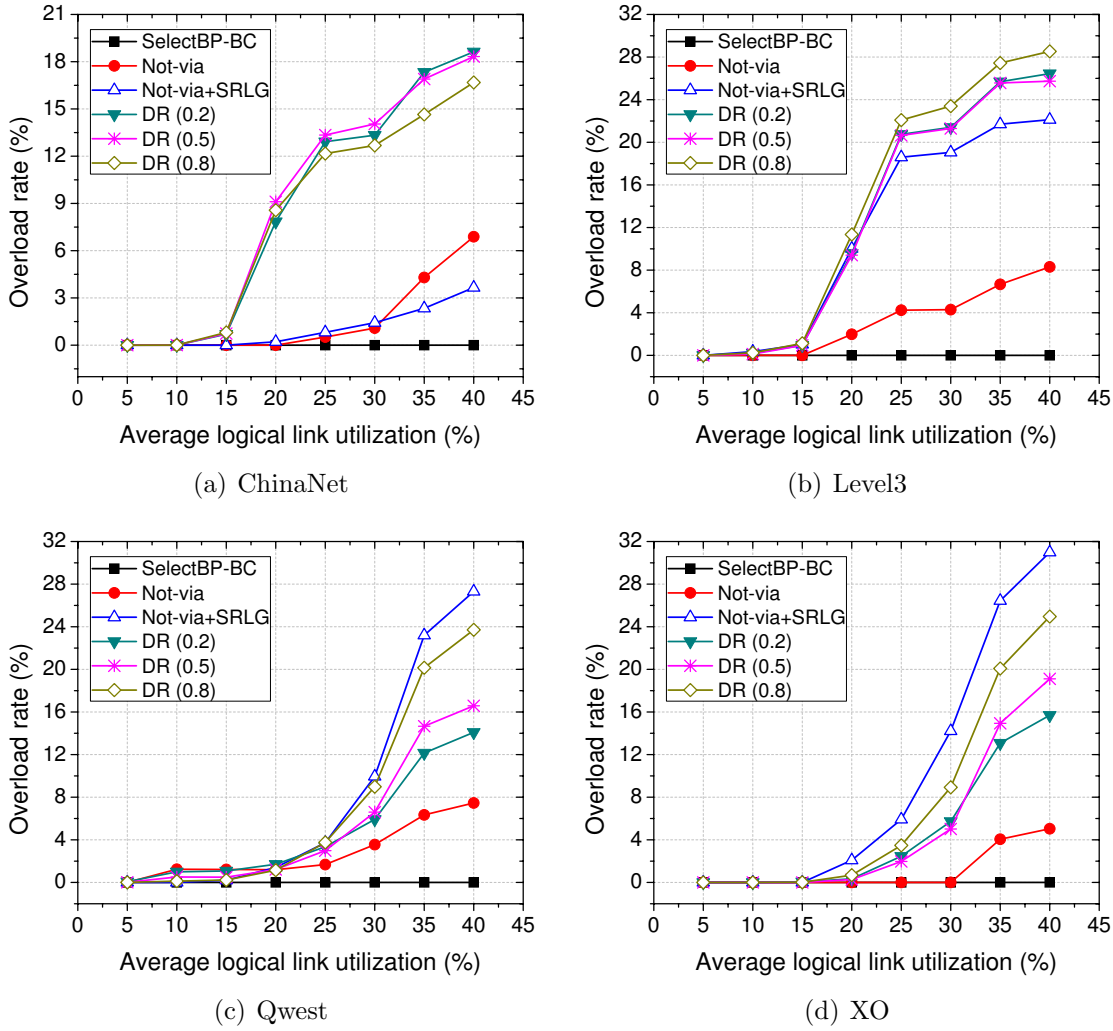


Figure 5.6. The overall traffic recovery rate for the four networks.

Next, we evaluate the overload rate under different logical link utilizations. The result in different networks is shown in Fig. 5.7, and the overall result for the four networks is shown in Fig. 5.8. SelectBP-BC avoids logical link overload with two techniques, i.e., using logical links with adequate bandwidth and controlling the rerouted traffic load. The other five algorithms may have quite high overload rate when the logical link utilization is above 15%.

We also measure the maximum logical link utilization on backup paths and show the result when the average logical link utilization is 40% in Table 5.6. By considering the bandwidth constraint, the maximum logical link utilization in SelectBP-BC is 100%; i.e., SelectBP-BC fully utilizes the bandwidth and does not cause logical link overload. The other five algorithms do not consider the bandwidth constraint when rerouting traffic, and then some logical links may be used by many backup paths at the same time. As a result, the maximum logical link utilization in these algorithms is quite high, which is even higher than 362%. Improperly rerouting traffic may cause severe congestion and strongly interferes with normal traffic. A survey to ISPs confirms the existence of severe congestion caused by IPFRR in their networks [40].



(a) ChinaNet (b) Level3

(c) Qwest (d) XO

Figure 5.7. The overload rate under different logical link utilizations.

5.4 Protect Each IP Link With Multiple Backup Paths

This section presents our second cross-layer approach which protects each IP link with multiple backup paths. We first describe the PCF model in Section 5.4.1, and then introduce an algorithms for selecting backup paths in Section 5.4.2. Finally, Section 5.4.3 presents the performance evaluation. Table 5.7 summarizes the notations used in this section.

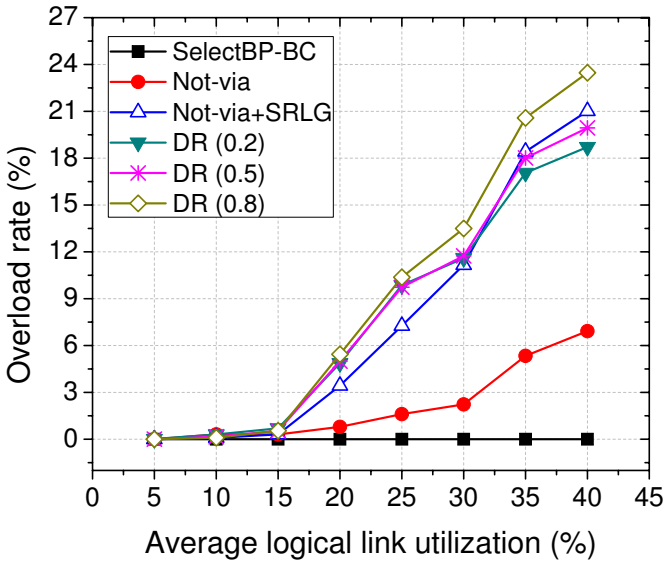


Figure 5.8. The overall overload rate for the four networks.

Table 5.6. The maximum logical link utilization on backup paths when the average logical link utilization is 40%

Algorithm	ChinaNet	Level3	Qwest	XO
SelectBP	100	100	100	100
Not-via	165.2	204	341.3	136.4
Not-via+SRLG	228.4	362.8	305.2	252
DR (0.2)	210	279.2	241.3	318.4
DR (0.5)	309	330.4	343.3	275.2
DR (0.8)	212.8	297.2	244.4	260.8

5.4.1 Probabilistically Correlated Failure Model

We first provide some motivation and then present the details of this model. Finally, we provide an example to show how the model works and differentiate it from the traditional independent and SRLG models.

5.4.1.1 Motivation

Recent Internet measurements [1, 2] show that there are two types of IP link failures in the Internet, i.e., *independent failures* and *correlated failures*. Independent failures have no relation. They occur for several reasons, such as hardware failures, configuration errors, and software bugs. Correlated failures are caused by failures

Table 5.7. Table of notations

Symbols	Meaning
	Section 5.4.1
$f_{i,j}$	the fiber link from $v_i \in V_P$ to $v_j \in V_P$
$e_{m,n}$	the logical link from $v_m \in V_L$ to $v_n \in V_L$
$p_{m,n}$	failure probability of $e_{m,n}$, where $p_{m,n} \in [0, 1)$
$q_{i,j}$	failure probability of $f_{i,j}$, where $q_{i,j} \in [0, 1)$
$a_{i,j}^{m,n}$	1 if $e_{m,n}$ is embedded on $f_{i,j}$, 0 otherwise
$F_{m,n}$	fiber links shared by $e_{m,n}$ and other logical links
$p_{m,n}^{C}$	correlated failure probability of $e_{m,n}$
$p_{m,n}^{I}$	independent failure probability of $e_{m,n}$
$P(f_{i,j}\|e_{m,n})$	failure probability of $f_{i,j}$ when $e_{m,n}$ fails
$P(e_{s,t}\|e_{m,n})$	failure probability of $e_{s,t}$ when $e_{m,n}$ fails
$B_{m,n}^{k}$	the kth backup path of $e_{m,n}$
$P(B_{m,n}^{k}\|e_{m,n})$	failure probability of $B_{m,n}^{k}$ when $e_{m,n}$ fails
$x_{s,t}^{m,n,k}$	1 if $B_{m,n}^{k}$ traverses $e_{s,t}$, 0 otherwise
	Section 5.4.2
N	each logical link has at most N backup paths
$c_{m,n}$	capacity of $e_{m,n}$
$l_{m,n}$	traffic load on $e_{m,n}$ under normal conditions
$r_{m,n}^{k}$	rerouted traffic load on $B_{m,n}^{k}$
$D_{m,n}$	traffic disruption of $e_{m,n}$
D	routing disruption of the entire network

of the fiber links carrying multiple logical links. When a logical link has a correlated failure, it implies that some other logical links sharing fiber links with it may also fail.

When a logical link fails, routers cannot determine whether it is an independent failure or a correlated failure. Therefore, the failure of $e_{m,n}$ implies that the logical links sharing at least one fiber link with $e_{m,n}$ may also fail with a certain probability. This probabilistic correlation between logical link failures cannot be modeled by the traditional independent and SRLG models as a *none-or-all* relation. The independent model considers that logical links only have independent failures; whereas the SRLG model considers that logical links only have correlated failures.

We develop a PCF model to deal with the probabilistic correlation between logical link failures. The objective is to quantify the impact of a logical link failure

on the failure probability of other logical links and backup paths. Based on this model, we can choose reliable paths to protect logical links.

5.4.1.2 The PCF Model

The PCF model is built on three kinds of information, i.e., the topology mapping, failure probability of fiber links, and failure probability of logical links, all of which are already gathered by ISPs. ISPs configure their topology mapping, and thus they have this information. The failure probability of fiber links and logical links can be obtained with Internet measurement approaches [1, 2] deployed at the optical and IP layers. Monitoring mechanisms at the optical layer can detect fiber link failures through SONET alarms. The information of logical link failures can be extracted from routing updates. ISPs also maintain failure information, because they monitor the optical and IP layers of their networks.

A key observation is that the backup paths for logical link $e_{m,n}$ are used only when $e_{m,n}$ fails. Therefore, the failure probability of these backup paths should be computed under the condition that $e_{m,n}$ fails. A backup path is built on logical links, and a logical link is embedded on fiber links. Hence, we first compute the failure probability of fiber links under the condition that $e_{m,n}$ fails. Then, we compute the failure probability of logical links and backup paths.

The failure probability of logical link $e_{m,n}$ is denoted by $p_{m,n} \in [0, 1)$. Unlike logical links, most fiber link failures are independent [37, 84]. We assume that a fiber link $f_{i,j}$ fails independently with probability $q_{i,j} \in [0, 1)$. In practice, we may obtain $p_{m,n}$ and $q_{i,j}$ based on previous logical link failures and fiber link failures. Let $a_{i,j}^{m,n}$ express the mapping between logical link $e_{m,n}$ and fiber link $f_{i,j}$.

$$a_{i,j}^{m,n} = \begin{cases} 1 & \text{if } e_{m,n} \text{ is embedded on } f_{i,j} \\ 0 & \text{otherwise} \end{cases} \tag{5.23}$$

A logical link has independent failures and correlated failures. Correlated failures are caused by the fiber links carrying multiple logical links. Let $F_{m,n}$ be the fiber links shared by $e_{m,n}$ and other logical links. $F_{m,n}$ is defined by Eq. (5.24).

$$F_{m,n} = \{f_{i,j} | a_{i,j}^{m,n} a_{i,j}^{s,t} = 1, \exists e_{s,t} \in E_L, \forall f_{i,j} \in F_P\} \tag{5.24}$$

Let $p^C_{m,n}$ be the probability that $e_{m,n}$ has correlated failures with other logical links. Since fiber link failures are independent, $p^C_{m,n}$ is computed by Eq. (5.25).

$$p^C_{m,n} = \begin{cases} 0 & \text{if } F_{m,n} = \emptyset \\ 1 - \prod_{f_{i,j} \in F_{m,n}} (1 - q_{i,j}) & \text{otherwise} \end{cases} \tag{5.25}$$

Suppose the independent failure probability of $e_{m,n}$ is $p^I_{m,n}$. We have the relation shown in Eq. (5.26), because failures of $e_{m,n}$ are either independent or correlated. Lemma 1 shows the range of $p^I_{m,n}$.

$$p_{m,n} = 1 - (1 - p^I_{m,n})(1 - p^C_{m,n}) \tag{5.26}$$

Lemma 1. $p^I_{m,n} \in [0, 1)$

Proof. According to Eq. (5.26), $p^I_{m,n} = \frac{p_{m,n} - p^C_{m,n}}{1 - p^C_{m,n}}$. Since $q_{i,j} \in [0, 1)$ for each fiber link $q_{i,j}$, we have $p^C_{m,n} \in [0, 1)$. Together with $p_{m,n} \in [0, 1)$, we have $p^I_{m,n} < 1$. Therefore, $p^I_{m,n} \in [0, 1)$. □

Based on the above information, we compute the failure probability of fiber links under the condition that $e_{m,n}$ fails. Let $P(f_{i,j}|e_{m,n})$ be this conditional failure probability. We only need to deal with the fiber link $f_{i,j}$ with failure probability $q_{i,j} > 0$. There are three cases as follows.

- Case 1: $e_{m,n}$ is not embedded on $f_{i,j}$. It means that the failure of $e_{m,n}$ is not related with $f_{i,j}$. Hence, $P(f_{i,j}|e_{m,n})$ is equal to $q_{i,j}$.

- Case 2: $f_{i,j}$ only carries $e_{m,n}$, then a failure of $f_{i,j}$ leads to an independent failure of $e_{m,n}$. In this case, $P(f_{i,j}|e_{m,n})$ is calculated by Eq. (5.27), where $P(e^I_{m,n}|e_{m,n})$ is the probability that $e_{m,n}$ has an independent failure when it fails, and $P(f_{i,j}|e^I_{m,n})$ is the probability that this independent failure is caused by $f_{i,j}$.

$$P(f_{i,j}|e_{m,n}) = P(e^I_{m,n}|e_{m,n}) \times P(f_{i,j}|e^I_{m,n}) \tag{5.27}$$

Based on Bayes' theorem, $P(e^I_{m,n}|e_{m,n})$ is calculated by Eq. (5.28). $P(e_{m,n}|e^I_{m,n})$ is the failure probability of $e_{m,n}$ when its independent failures occur, which is

1. $P(e^I_{m,n})$ is the independent failure probability of $e_{m,n}$, i.e., $p^I_{m,n}$. $P(e_{m,n})$ is the failure probability of $e_{m,n}$, i.e., $p_{m,n}$.

$$P(e^I_{m,n}|e_{m,n}) = \frac{P(e_{m,n}|e^I_{m,n})P(e^I_{m,n})}{P(e_{m,n})} = \frac{P(e^I_{m,n})}{P(e_{m,n})} = \frac{p^I_{m,n}}{p_{m,n}} \tag{5.28}$$

Similarly, $P(f_{i,j}|e^I_{m,n})$ is computed by Eq. (5.29). Since $q_{i,j}$ is not 0, $p^I_{m,n}$ cannot be 0.

$$P(f_{i,j}|e^I_{m,n}) = \frac{P(e^I_{m,n}|f_{i,j})P(f_{i,j})}{P(e^I_{m,n})} = \frac{q_{i,j}}{p^I_{m,n}} \tag{5.29}$$

Based on Eq. (5.27)–Eq. (5.29), $P(f_{i,j}|e_{m,n}) = \frac{q_{i,j}}{p_{m,n}}$.

- Case 3: $f_{i,j}$ carries $e_{m,n}$ and some other logical links, then a failure of $f_{i,j}$ leads to correlated failures. The calculation of $P(f_{i,j}|e_{m,n})$ is similar to that in case 2, in which $e^I_{m,n}$ is replaced by $e^C_{m,n}$ and $p^I_{m,n}$ is replaced by $p^C_{m,n}$. The result is the same as in case 2, i,e., $P(f_{i,j}|e_{m,n}) = \frac{q_{i,j}}{p_{m,n}}$.

In summary, the failure probability $P(f_{i,j}|e_{m,n})$ is given by Eq. (5.30). It is only defined for $e_{m,n}$ whose failure probability $p_{m,n}$ is not 0. If $p_{m,n}$ is 0, $e_{m,n}$ never fails, and thus we do not need to select backup paths for it.

$$P(f_{i,j}|e_{m,n}) = \begin{cases} q_{i,j} & a^{m,n}_{i,j} = 0 \\ \frac{q_{i,j}}{p_{m,n}} & a^{m,n}_{i,j} = 1 \end{cases} \tag{5.30}$$

Lemma 2. *$P(f_{i,j}|e_{m,n}) > q_{i,j}$, if $f_{i,j}$ carries $e_{m,n}$ and $q_{i,j} > 0$.*

Proof. If $f_{i,j}$ carries $e_{m,n}$, it is the second case of Eq. (5.30), i.e., $P(f_{i,j}|e_{m,n}) = \frac{q_{i,j}}{p_{m,n}}$. Since $q_{i,j} > 0$, the failure probability of $e_{m,n}$ is above 0. Since $p_{m,n} \in [0, 1)$, we have $p_{m,n} \in (0, 1)$. Therefore, $P(f_{i,j}|e_{m,n}) = \frac{q_{i,j}}{p_{m,n}} > q_{i,j}$. □

Lemma 2 indicates the condition under which the failure probability of $f_{i,j}$ is increased due to a failure of $e_{m,n}$. Accordingly, the failure probability of the logical links embedded on $f_{i,j}$ is affected by the failure of $e_{m,n}$.

Next, we calculate the failure probability of logical links under the condition that $e_{m,n}$ fails. A logical link $e_{s,t}$ has independent failures and correlated failures. Its independent failure probability is $p^I_{s,t}$, whether or not $e_{m,n}$ fails. However, its correlated failure probability may be affected by the failure of $e_{m,n}$. We use $P(e^C_{s,t}|e_{m,n})$ to denote the correlated failure probability of $e_{s,t}$ under the condition that $e_{m,n}$ fails. It is calculated by Eq. (5.31).

$$P(e^C_{s,t}|e_{m,n}) = \begin{cases} 0 & \text{if } F_{m,n} = \emptyset \\ 1 - \prod_{f_{i,j} \in F_{s,t}} (1 - P(f_{i,j}|e_{m,n})) & \text{otherwise} \end{cases} \tag{5.31}$$

Based on this, Eq. (5.32) computes the failure probability of $e_{s,t}$ under the condition that $e_{m,n}$ fails, which is denoted by $P(e_{s,t}|e_{m,n})$.

$$P(e_{s,t}|e_{m,n}) = 1 - (1 - p^I_{s,t})(1 - P(e^C_{s,t}|e_{m,n})) \tag{5.32}$$

Theorem 9. *$P(e_{s,t}|e_{m,n}) > p_{s,t}$, if there is a fiber link $f_{i,j}$ satisfying $a^{s,t}_{i,j} = 1$, $a^{m,n}_{i,j} = 1$, and $q_{i,j} > 0$.*

Proof. Since $p^I_{s,t} \in [0, 1)$ as shown by Lemma 1, $1 - p^I_{s,t} \in (0, 1]$. According to Eq. (5.26) and Eq. (5.32), we need to prove $P(e^C_{s,t}|e_{m,n}) > p^C_{s,t}$ to show $P(e_{s,t}|e_{m,n}) > p_{s,t}$. With Eq. (5.25) and Eq. (5.31), it is equivalent to proving the following relation.

$$\prod_{f_{i,j} \in F_{m,n}} (1 - P(f_{i,j}|e_{m,n})) < \prod_{f_{i,j} \in F_{m,n}} (1 - q_{i,j})$$

If $q_{i,j}$ is 0, $P(f_{i,j}|e_{m,n})$ is also 0, and thus $1 - P(f_{i,j}|e_{m,n}) = 1$ and $1 - q_{i,j} = 1$. If $f_{i,j}$ carries $e_{m,n}$ and $q_{i,j} > 0$, Lemma 2 shows that $P(f_{i,j}|e_{m,n}) > q_{i,j}$. Therefore, $\prod_{f_{i,j} \in F_{m,n}} (1 - P(f_{i,j}|e_{m,n})) < \prod_{f_{i,j} \in F_{m,n}} (1 - q_{i,j})$, if there is a fiber link $f_{i,j}$ shared by $e_{m,n}$ and $e_{s,t}$ and $q_{i,j} > 0$. It means that $P(e_{s,t}|e_{m,n}) > p_{s,t}$. □

As shown by Theorem 9, if logical link $e_{s,t}$ shares a fiber link $f_{i,j}$ with $e_{m,n}$ and $q_{i,j}$ is not 0, the failure probability of $e_{s,t}$ is increased due to the failure of $e_{m,n}$. Accordingly, the failure probability of the backup paths built on $e_{s,t}$ is increased when $e_{m,n}$ fails.

Then, we calculate the failure probability of backup paths under the condition that $e_{m,n}$ fails. Let $B^k_{m,n}$ denote the kth backup path of $e_{m,n}$, and $P(B^k_{m,n}|e_{m,n})$ denote the failure probability of $B^k_{m,n}$ when $e_{m,n}$ fails. The variable $x^{m,n,k}_{s,t} \in \{0,1\}$ is defined as follows to express if $B^k_{m,n}$ traverses the logical link $e_{s,t}$.

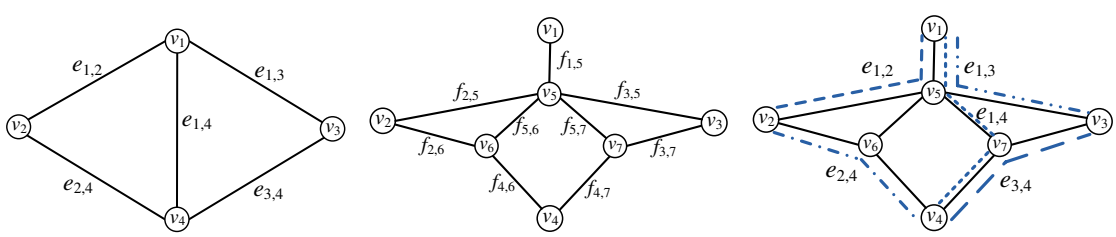
$$x^{m,n,k}_{s,t} = \begin{cases} 1 & \text{if } B^k_{m,n} \text{ traverses } e_{s,t} \\ 0 & \text{otherwise} \end{cases} \tag{5.33}$$

$P(B^k_{m,n}|e_{m,n})$ is computed by Eq. (5.34), which enumerates logical links and counts the ones traversed by $B_{m,n}$.

$$P(B^k_{m,n}|e_{m,n}) = 1 - \prod_{e_{s,t} \in E_L} (1 - x^{m,n,k}_{s,t} P(e_{s,t}|e_{m,n})) \tag{5.34}$$

5.4.1.3 An Example

We use an example to show how the PCF model works and the differences from the independent and SRLG models. The physical topology, logical topology, and topology mapping are shown in Fig. 5.9. For simplicity, fiber links have the same failure probability 0.01. The failure probability of logical links is shown in Table 5.8.



(a) Logical topology. (b) Physical topology. (c) The mapping between the logical and fiber links.

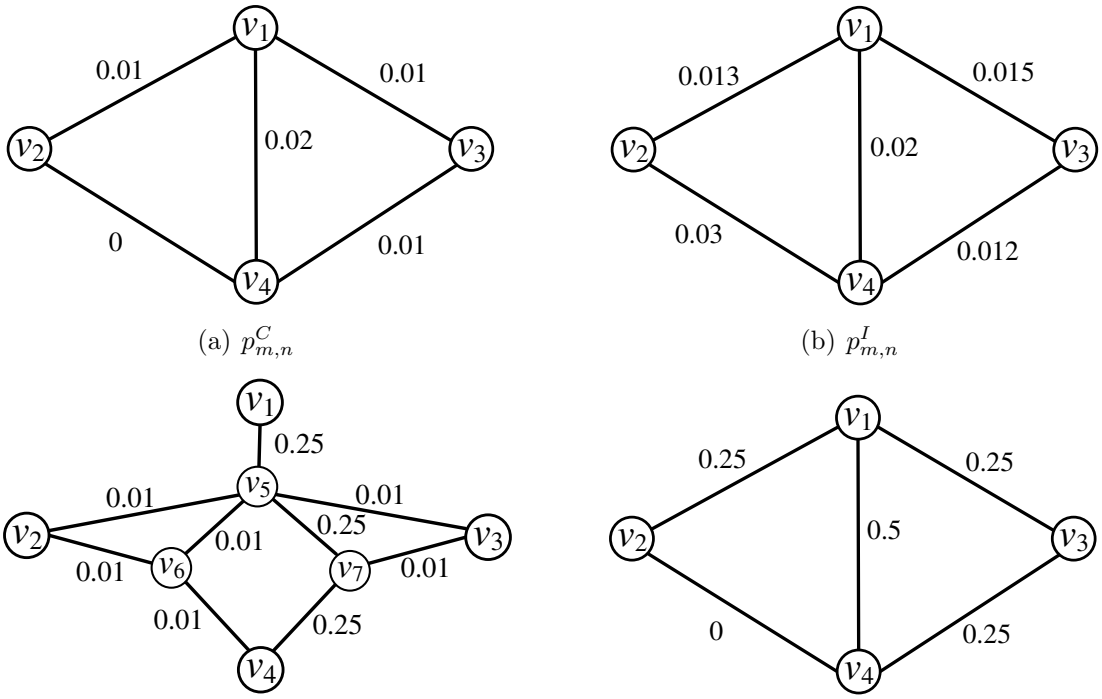
Figure 5.9. An example of the topology mapping.

Table 5.8. The failure probability of logical links

$e_{1,2}$	$e_{1,3}$	$e_{1,4}$	$e_{2,4}$	$e_{3,4}$
0.023	0.025	0.040	0.030	0.022

Suppose v_1 needs to protect logical link $e_{1,4}$. First, Eq. (5.25) calculates the correlated failure probability of logical links. The result is shown in Fig. 5.10(a).

For example, two fiber links are shared by $e_{1,4}$ and other logical links. Hence, set $F_{1,4} = \{f_{1,5}, f_{4,7}\}$, and $p^C_{1,4} = 1 - (1 - q_{1,5})(1 - q_{4,7}) = 1 - 0.99^2 = 0.02$. With Eq. (5.26), we can compute the independent failure probability of logical links, which is shown in Fig. 5.10(b). Then, we use Eq. (5.30) to compute the failure probability of fiber links under the condition that $e_{1,4}$ fails. The result is shown in Fig. 5.10(c). Next, Eq. (5.31) computes correlated failure probability $P(e^C_{s,t}|e_{1,4})$ and the result is shown in Fig. 5.10(d). Note that, $P(e^C_{1,4}|e_{1,4}) = \frac{p^C_{1,4}}{p_{1,4}} = 0.5$. Based on it, Eq. (5.32) computes the failure probability $P(e_{s,t}|e_{1,4})$, which is shown in the last row of Table 5.9.



(a) $p^C_{m,n}$ (b) $p^I_{m,n}$

(c) $P(f_{i,j}|e_{1,4})$ (d) $P(e^C_{s,t}|e_{1,4})$

Figure 5.10. An example of calculating failure probability with the PCF model.

Table 5.9 summarizes the failure probability $P(e_{s,t}|e_{1,4})$ computed by the three models. The models lead to different results for $e_{1,2}$, $e_{1,3}$, and $e_{3,4}$, because these logical links share fiber links with $e_{1,4}$. Due to the omission of topology mapping, the independent model considers that logical link failures are independent, and hence it underestimates $P(e_{1,2}|e_{1,4})$, $P(e_{1,3}|e_{1,4})$, and $P(e_{3,4}|e_{1,4})$. The SRLG model overestimates the failure probability, because it considers that the logical links

sharing fiber links with $e_{1,4}$ all fail.

Table 5.9. $P(e_{s,t}|e_{1,4})$ computed by the three models

Model name	$e_{1,2}$	$e_{1,3}$	$e_{1,4}$	$e_{2,4}$	$e_{3,4}$
Independent	0.023	0.025	1	0.030	0.022
SRLG	1	1	1	0.030	1
PCF	0.260	0.261	1	0.030	0.259

Finally, we use Eq. (5.34) to compute the failure probability of backup paths. There are two loop-free paths that can be used as backup paths for $e_{1,4}$. Their failure probability computed by three models are shown in Table 5.10. The independent model underestimates the failure probability and considers that $B_{1,2}$ is more reliable than $B_{1,1}$. The SRLG model considers that both paths are disconnected. As the PCF model shows, the failure of $e_{1,4}$ affects the reliability of these two paths, and $B_{1,1}$ is more reliable than $B_{1,2}$ when $e_{1,4}$ fails.

Table 5.10. The failure probability of backup paths for $e_{1,4}$ computed by the three models.

Model name	$B_{1,1} : v_1 \rightarrow v_2 \rightarrow v_4$	$B_{1,2} : v_1 \rightarrow v_3 \rightarrow v_4$
Independent	0.052	0.046
SRLG	1	1
PCF	0.282	0.452

5.4.2 Backup Path Selection

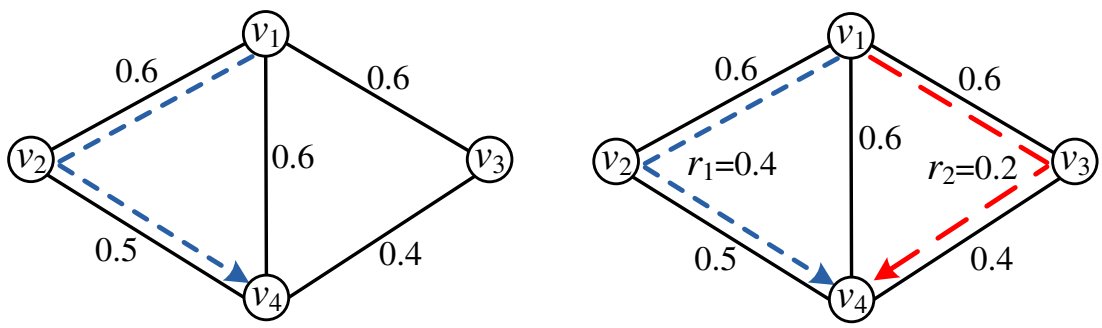
Based on the PCF model, we propose an algorithm to select multiple backup paths to protect each IP link. Our algorithm considers both reliability and bandwidth constraints. It aims at minimizing routing disruption by choosing reliable backup paths and splitting the rerouted traffic onto them. Furthermore, it controls the rerouted traffic load to prevent causing logical link overload.

5.4.2.1 Motivation

Most existing protection approaches focus on choosing reliable backup paths, but ignore the fact that a backup path may not have enough bandwidth for the rerouted traffic. Without considering the bandwidth constraint, they commonly choose one

backup path to protect each logical link. Our approach considers both reliability and bandwidth constraint. It protects each logical link with multiple backup paths and splits the rerouted traffic onto them, because there may be no individual backup path that has enough bandwidth for the rerouted traffic.

Fig. 5.11 shows the motivation for protecting a logical link with multiple backup paths. Logical links have bandwidth 1, and the number on a logical link is its traffic load under normal conditions. In Fig. 5.11(a), v_1 uses single backup path, whose usable bandwidth is $\min\{1-0.6, 1-0.5\} = 0.4$. When $e_{1,4}$ fails, the total traffic load on $e_{1,2}$ will exceed its bandwidth, and hence link overload occurs. Our approach protects $e_{1,4}$ with two backup paths as shown in Fig. 5.11(b). When $e_{1,4}$ fails, the rerouted traffic load split onto the left one is 0.4, and that onto the right one is 0.2. With this approach, the entire traffic of $e_{1,4}$ can be rerouted without causing link overload.



(a) Single backup path may not have enough bandwidth.

(b) The rerouted traffic is split on two backup paths.

Figure 5.11. Motivation for protecting a logical link with multiple backup paths.

Using more backup paths to protect a logical link is helpful for rerouting traffic, but increases configuration complexity and storage overhead. We require that each logical link can have at most N backup paths. In Section 5.4.3, we will investigate the impact of parameter N on the performance of our approach.

5.4.2.2 Problem Definition and Formulation

We consider both reliability of backup paths and bandwidth constraint of logical links. The objective is to minimize the routing disruption of the entire network, which was also the major objective in prior works. Furthermore, the rerouted traffic

load on a logical link should not exceed its usable bandwidth to prevent logical link overload and interfering with normal traffic. This constraint is ignored by existing approaches. The backup path selection problem is described in Definition 4.

Definition 4 (Problem definition)**.** *We aim at selecting at most N backup paths for each logical link and computing the rerouted traffic load for each backup path, such that (1) the routing disruption of the entire network is minimized; (2) the rerouted traffic load on each logical link does not exceed its usable bandwidth.*

First, we define routing disruption based on the PCF model. Suppose the capacity of logical link $e_{m,n}$ is $c_{m,n}$. Under normal conditions, the traffic load on $e_{m,n}$ is $l_{m,n}$ which satisfies $l_{m,n} \leq c_{m,n}$. Network administrators configure the link cost to achieve traffic engineering goals, and hence traffic load $l_{m,n}$ is known. The bandwidth of $e_{m,n}$ that can be used by backup paths is $c_{m,n} - l_{m,n}$. $e_{m,n}$ is *overloaded* if the rerouted traffic load on it exceeds $c_{m,n} - l_{m,n}$. The kth backup path is denoted by $B^k_{m,n}$, and the bandwidth reserved for it is $r^k_{m,n}$, which is the traffic load split onto $B^k_{m,n}$ when $e_{m,n}$ fails. The traffic load of $e_{m,n}$ protected by backup paths is $\sum_{k=1}^{N} r^k_{m,n}$, and the unprotected traffic load is $l_{m,n} - \sum_{k=1}^{N} r^k_{m,n}$. When $e_{m,n}$ fails, the unprotected traffic is disrupted. If $B^k_{m,n}$ also fails, the traffic rerouted on it is disrupted. Therefore, the traffic disruption of $e_{m,n}$ is defined in Eq. (5.35), which is the mathematical expectation of disrupted traffic load of $e_{m,n}$.

$$D_{m,n} = p_{m,n}(\sum_{k=1}^{N} P(B^k_{m,n}|e_{m,n}) r^k_{m,n} + l_{m,n} - \sum_{k=1}^{N} r^k_{m,n}) \tag{5.35}$$

The routing disruption of the entire network is then defined in Eq. (5.36), which is the mathematical expectation of traffic disruption in the entire network.

$$D = \sum_{e_{m,n} \in E_L} D_{m,n} \tag{5.36}$$

Our problem can be formally defined as an optimization problem as shown in Eq. (5.37)–Eq. (5.41), in which $x^{m,n,k}_{s,t}$ and $r^k_{m,n}$ are unknown variables. The connectivity constraint in Eq. (5.38) requires that the selected logical links form paths. The constraint in Eq. (5.39) requires that the rerouted traffic load on each logical link does not exceed its usable bandwidth. Eq. (5.40) specifies that the

rerouted traffic load on the backup paths for $e_{m,n}$ does not exceed $l_{m,n}$. Ideally, all the traffic load of $e_{m,n}$ should be rerouted. In some cases, the network does not have enough bandwidth, and hence the overall rerouted traffic load of $e_{m,n}$ may be lower than $l_{m,n}$.

$$\text{minimize} \quad D \tag{5.37}$$

subject to

$$\sum_{j:e_{i,j}\in E_L} x_{i,j}^{m,n,k} - \sum_{j:e_{j,i}\in E_L} x_{j,i}^{m,n,k} = \begin{cases} 1 & i = m \\ -1 & i = n \\ 0 & \text{otherwise} \end{cases} \tag{5.38}$$

$$\sum_{e_{m,n}\in E_L} \sum_{k=1}^{N} x_{i,j}^{m,n,k} r_{m,n}^{k} \leq c_{i,j} - l_{i,j} \tag{5.39}$$

$$\sum_{k=1}^{N} r_{m,n}^{k} \leq l_{m,n} \tag{5.40}$$

$$\forall e_{m,n}, e_{i,j} \in E_L, \quad 1 \leq k \leq N, \quad x_{i,j}^{m,n,k} \in \{0,1\}, \quad 0 \leq r_{m,n}^{k} \leq l_{m,n} \tag{5.41}$$

5.4.2.3 An Algorithm

The above formulation is a mixed integer nonlinear programming problem which is NP-hard. We propose a multi-round algorithm to solve the problem. The basic idea is to select backup paths one by one, until there is no usable bandwidth or no any logical link can have more backup paths. The algorithm is shown in Algorithm 7. We use $D_{m,n}$ defined in Eq. (5.35) as the weight of $e_{m,n}$. In each round, the algorithm picks out the logical link $e_{m,n}$ with the largest weight. Suppose $e_{m,n}$ already has $k-1$ backup paths. Adding one more backup path reduces $D_{m,n}$ by

$\Delta_{m,n}$ shown in Eq. (5.42).

$$\Delta_{m,n} = p_{m,n} r^k_{m,n} (1 - P(B^k_{m,n} | e_{m,n})) \tag{5.42}$$

Our strategy is to choose $B^k_{m,n}$ and determine $r^k_{m,n}$ to maximize $\Delta_{m,n}$. We develop Algorithm 8 to achieve this. The basic idea is similar to the Dijkstra's algorithm for calculating the shortest path. Starting from node v_i, the algorithm gradually adds logical links to expand the backup path. The rerouted traffic load is the smaller one between the unprotected traffic load of $e_{m,n}$ and the usable bandwidth of the backup path. The computational complexity of Algorithm 8 is $O((|E_L| + |V_L|) \log(|V_L|))$.

Algorithm 7 invokes Algorithm 8 to choose a backup path for $e_{m,n}$ and then determines the rerouted traffic load (lines 12–19). Next, it updates the usable bandwidth (lines 20–25), number of backup paths (line 26), unprotected traffic load (line 27), and weight of $e_{m,n}$ (line 28). The algorithm removes a logical link from the queue, if it has N backup paths or its traffic is fully protected (lines 29–31). It terminates when the queue is empty. The computational complexity of Algorithm 7 is $O(N|E_L|(|E_L| + |V_L|) \log(|V_L|))$ in the worst case, when each logical link has N backup paths.

5.4.3 Performance Evaluation

We evaluate the performance of the proposed approach and compare it with other backup path-based approaches.

5.4.3.1 Simulation Setup

The network topologies, IP layer configuration, and algorithms compared with our approach are the same as Section 5.3.3. The link failure probability is set based on the Internet measurement [1]. We randomly choose 2.5% logical links to be high failure links. Their failure probability is between 0.5% and 0.1% and satisfies the power-law distribution with parameter -0.73. For other logical links, their failure probability is between 0.1% and 0.01% and satisfies the power-law distribution with parameter -1.35. The failure probability of fiber links is between 0.05% and 0.01%

Algorithm 7 SelectBP-MP

Procedure:

```
Initialize a priority queue Q
for each logical link e_{m,n} ∈ E_L do
    w_{m,n} ← p_{m,n} l_{m,n}    ▷ the weight assigned to w_{m,n}
    b_{m,n} ← c_{m,n} − l_{m,n}    ▷ the usable bandwidth of e_{m,n}
    n_{m,n} ← 0    ▷ the number of backup paths for e_{m,n}
    u_{m,n} ← l_{m,n}    ▷ the unprotected traffic load of e_{m,n}
    ENQUEUE(Q, e_{m,n})
end for
while Q ≠ ∅ do
    e_{m,n} ← the logical link in Q with the largest weight
    k ← n_{m,n} + 1
    B^k_{m,n} ← run MaxWeightPath on G_L for e_{m,n}
    if B^k_{m,n} does not exist then
        DEQUEUE(Q, e_{m,n})
    else
        r^k_{m,n} ← the usable bandwidth of B^k_{m,n}
        if u_{m,n} < r^k_{m,n} then
            r^k_{m,n} ← u_{m,n}
        end if
        for each logical link e_{s,t} on B^k_{m,n} do
            b_{s,t} ← b_{s,t} − r^k_{m,n}
            if b_{s,t} = 0 then
                G_L ← G_L − e_{s,t}    ▷ e_{s,t} does not have usable bandwidth
            end if
        end for
        c_{m,n} ← c_{m,n} + 1
        u_{m,n} ← u_{m,n} − r^k_{m,n}    ▷ update the unprotected traffic load
        w_{m,n} ← w_{m,n} − p_{m,n} r^k_{m,n} (1 − P(B^k_{m,n} | e_{m,n}))
        if c_{m,n} = N or u_{m,n} = 0 then
            DEQUEUE(Q, e_{m,n})
        end if
    end if
end while
```

and satisfies the power-law distribution with parameter -1. In the simulation, we use 100 failure probability settings and each simulation is run 1,000 times for each failure probability setting. Each time, we randomly choose one logical link and determine if it is an independent failure or a correlated failure based on the failure

Algorithm 8 MaxWeightPath

Procedure:

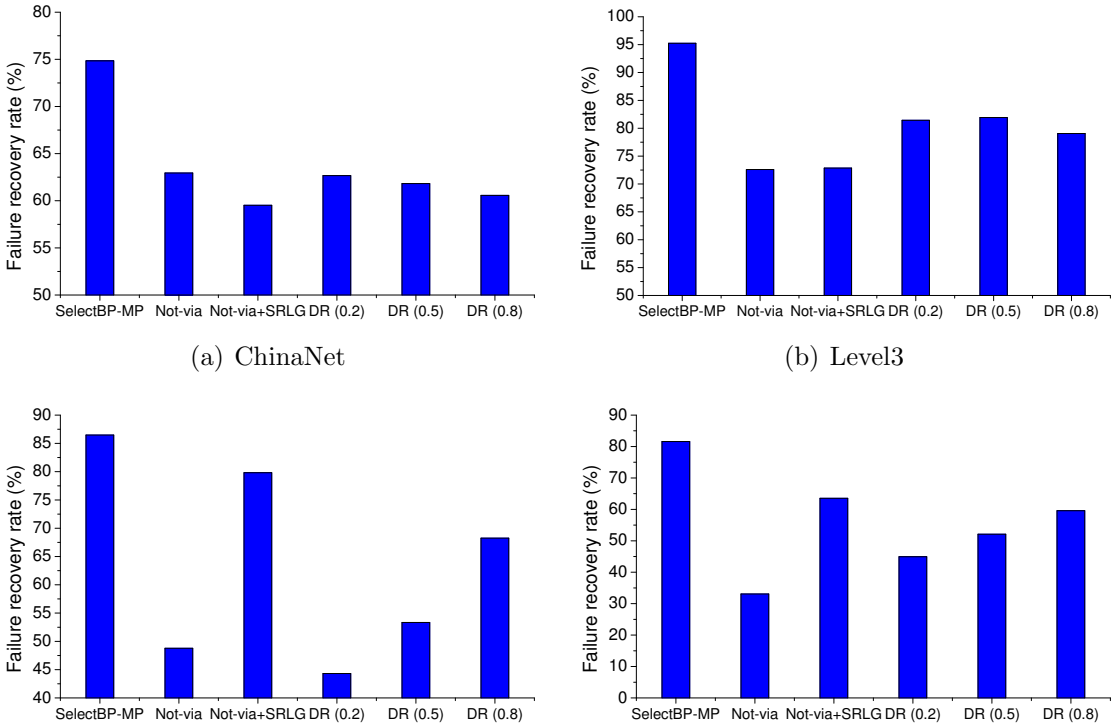
1: $G_L \leftarrow G_L - e_{m,n}$ ▷ the backup path should avoid $e_{m,n}$
2: Initialize a priority queue Q
3: **for** each node $v_s \in V_L$ **do**
4: $w(v_s) \leftarrow -\infty$
5: $visit(v_s) \leftarrow 0$
6: $prev(v_s) \leftarrow NULL$
7: ENQUEUE(Q, v_s)
8: **end for**
9: $w(v_i) \leftarrow 0$
10: **while** $Q \neq \emptyset$ **do**
11: $v_s \leftarrow$ the node in Q with the largest weight
12: **if** $w(v_s) = -\infty$ **then**
13: **break**
14: **end if**
15: $visit(v_s) \leftarrow 1$
16: **for** each neighbor v_t of v_s with $visit(v_t) = 0$ **do**
17: $B_{i,t} \leftarrow$ the path from v_i to v_t constructed from $prev$
18: $r_{i,t} \leftarrow$ unprotected traffic load of $e_{m,n}$
19: **if** $r_{i,t} >$ the usable bandwidth of $B_{i,t}$ **then**
20: $r_{i,t} \leftarrow$ the usable bandwidth of $B_{i,t}$
21: **end if**
22: **if** $w(v_t) < r_{i,t}(1 - P(B_{i,t}|e_{m,n}))$ **then**
23: $w(v_t) \leftarrow u_{i,t}(1 - P(B_{i,t}|e_{m,n}))$
24: $prev(v_t) \leftarrow v_s$
25: **end if**
26: **end for**
27: **end while**
28: Using $prev$ to construct the path from v_i to v_j

probability. If it is a correlated failure, we fail one fiber link which is shared by this logical link and other logical links. The logical links embedded on the failed fiber link all fail.

5.4.3.2 Reliability of Backup Paths

We first investigate the reliability of the backup paths. In our algorithm SelectBP-MP, the logical link capacity is set to infinite and the parameter N is 1. Hence, the algorithm chooses the backup path with the lowest failure probability for each

logical link. In a test case, if a failed logical link has a live backup path, this logical link failure can be recovered. The *failure recovery rate* is the percentage of the recovered logical link failures, and it is used as the performance metric.



(a) ChinaNet (b) Level3

(c) Qwest (d) XO

Figure 5.12. The average failure recovery rate.

The average failure recovery rate across 100,000 test cases is shown in Fig. 5.12. We highlight three features. First, our algorithm SelectBP-MP outperforms the other five algorithms in all four networks. This shows that the PCF model is effective for finding reliable backup paths. Not-via ignores the correlation between logical link failures, and thus backup paths may traverse some failed logical links. Not-via+SRLG may remove some useful logical links and even disconnect the topology. Consequently, some logical links may not have backup paths. Second, unlike SelectBP-MP, the performance of the other five algorithms is not consistent across the four networks. For example, Not-via+SRLG is the second best in Qwest and XO, while it is the worst in ChinaNet. Similarly, DR (0.8) is better than DR (0.2) and DR (0.5) in Qwest and XO, but it is worse than them in ChinaNet and Level3. Third, the parameter p strongly affects the performance of DR, and it

is difficult to choose an appropriate p to achieve good performance in all networks. The overall result for the four networks is shown in Fig. 5.13. On average, the reliability of the backup paths chosen by our approach is at least 18% higher that that achieved by the other five algorithms.

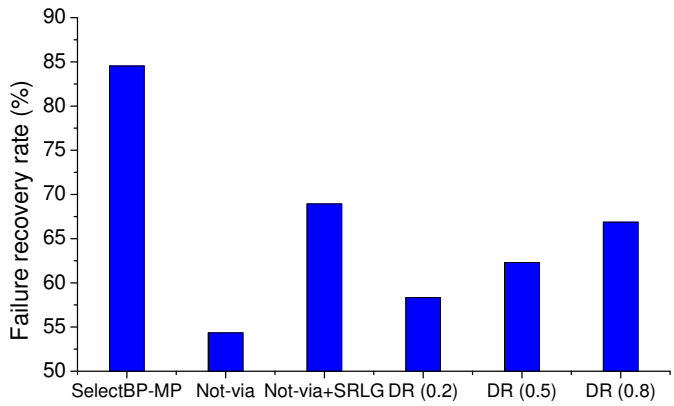


Figure 5.13. The overall failure recovery rate for the four networks.

5.4.3.3 Routing Disruption and Logical Link Overload

Next, we consider the traffic load and bandwidth constraint. The parameter N in the algorithm SelectBP-MP is varied from 1 to 5. We define the following two metrics for measuring the benefit and negative impact.

- *Routing disruption*: For a failed logical link $e_{m,n}$, if a backup path does not contain any failed or overloaded logical link, the traffic rerouted by it is recovered. Suppose the overall traffic load of failed logical links is T and the recovered traffic load is T_r, the routing disruption is defined as $\frac{T-T_r}{T}$. The optimal value (0%) means that no traffic is disrupted by failures.

- *Overload rate*: In a test case, we count the logical links traversed by the rerouted traffic and denote this number as L. We also count the overloaded ones among them. A logical link is overloaded if its capacity is smaller than the traffic load on it, including its own traffic and the rerouted traffic. Suppose there are L_o overloaded logical links. The overload rate is defined as $\frac{L_o}{L}$, and it quantifies the negative impact caused by the rerouted traffic.

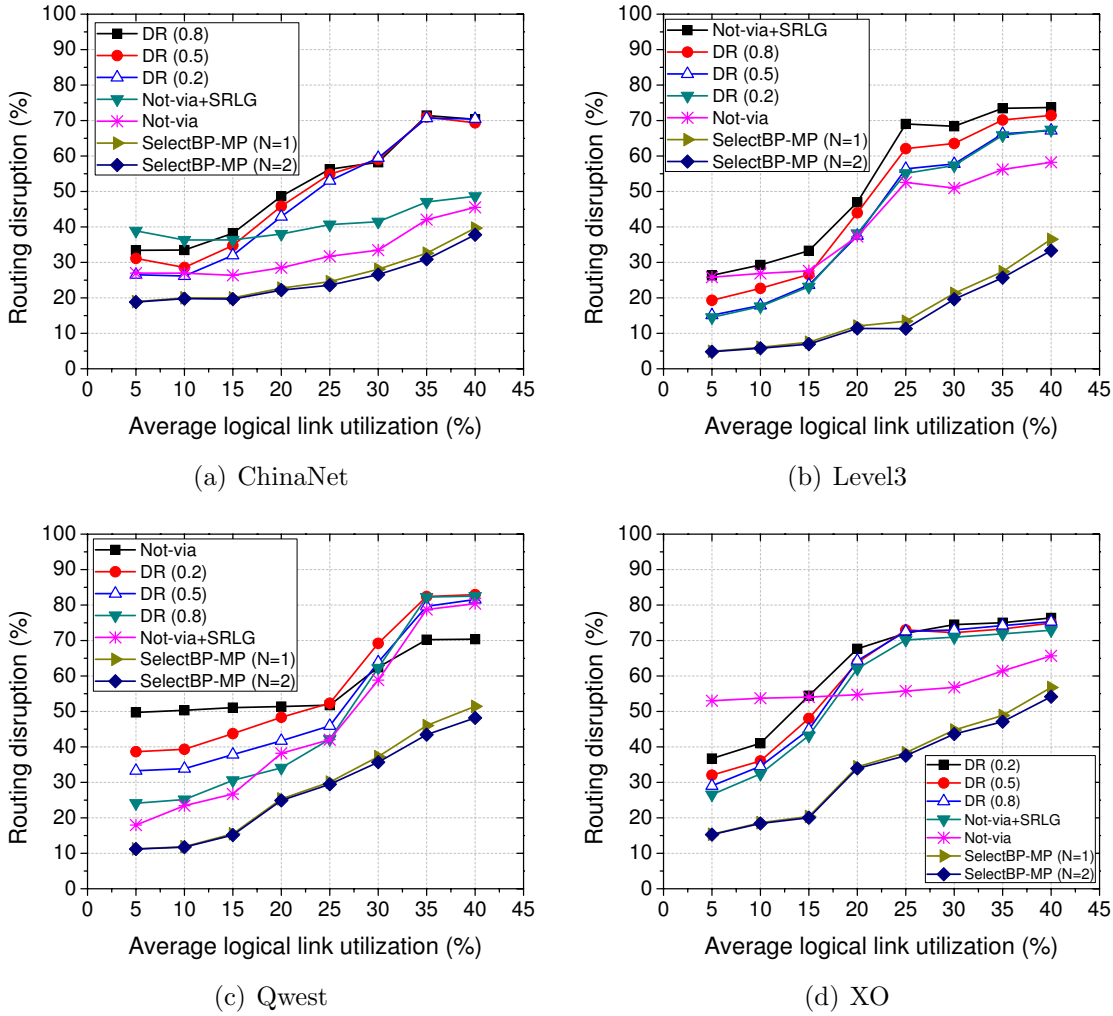


Figure 5.14. The average routing disruption under different logical link utilizations.

The average routing disruption is shown in Fig. 5.14. We highlight four important aspects of the simulation results. First, using more backup paths can reduce the routing disruption, especially when the logical link utilization is high. The performance of SelectBP-MP does not vary much when $N \geq 2$ and thus we only show the result when $N = 1$ and $N = 2$. It means that two backup paths should be adequate for protecting a logical link. Second, SelectBP-MP outperforms the other five algorithms under different logical link utilizations in each network. Third, the performance of the other five algorithms is not consistent across the four networks, which is similar to the failure recovery rate in Fig. 5.12. Our approach is better than the other five in adapting to different networks and logical link utilizations.

Fourth, the routing disruption increases as the logical link utilization increases due to lack of usable bandwidth. For example, if the logical link utilization is 25%, the usable bandwidth is 3 times the traffic load. However, when it increases to 40%, the usable bandwidth decreases to 1.5 times the traffic load. A small increase in the traffic load makes rerouting much more difficult. The overall result for the four networks is shown in Fig. 5.15. On average, the routing disruption of our approach is at least 22% lower that that of the other five algorithms.

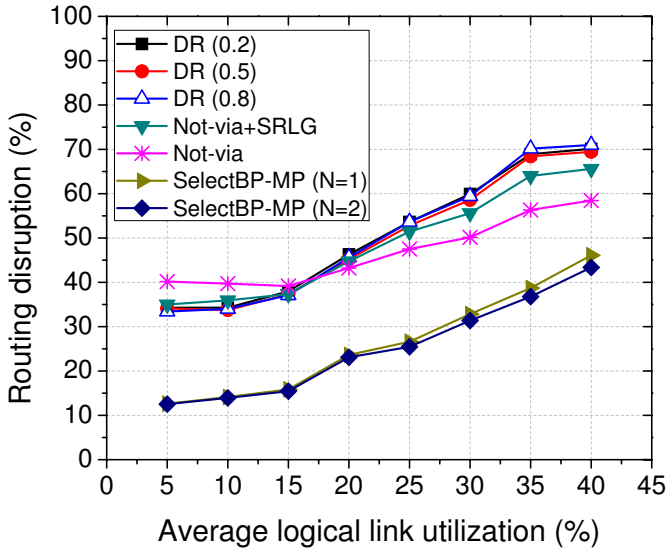
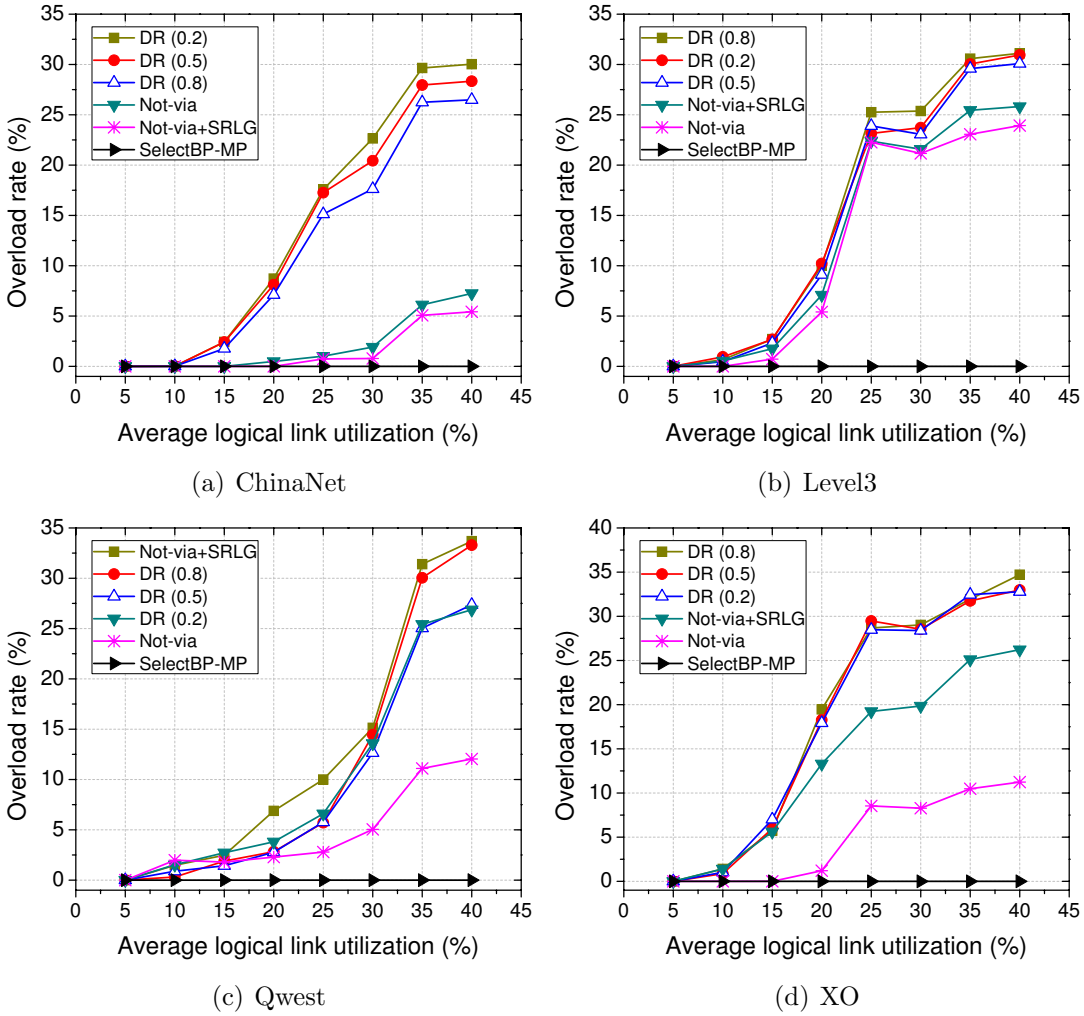


Figure 5.15. The overall routing disruption for the four networks.

Then, we evaluate the overload rate under different logical link utilizations, which is shown in Fig. 5.16. SelectBP-MP avoids logical link overload with two techniques, i.e., using logical links with usable bandwidth and controlling the rerouted traffic load. The other five algorithms may have quite high overload rate when the logical link utilization is above 20%.

We also measure the maximum logical link utilization on recovery paths and show the result when the average logical link utilization is 40% in Table 5.11. By considering the bandwidth constraint, the maximum logical link utilization in SelectBP-MP is 100%, which means that SelectBP-MP fully utilizes the bandwidth and does not cause logical link overload. The other five algorithms do not consider the bandwidth constraint when rerouting traffic, and hence some logical links may be used by many backup paths at the same time. As a result, the maximum logical



(a) ChinaNet (b) Level3

(c) Qwest (d) XO

Figure 5.16. The overload rate under different logical link utilizations.

link utilization in these algorithms is quite high.

5.5 Related Work

There are three categories of existing works that are related to our approach.

Backup path-based IP link protection: Most existing works consider the backup path selection as a connectivity problem and mainly focus on finding backup paths to bypass the failed IP links [32, 88, 36, 39]. Some other works address the backup path selection when IP links have different failure probability, and aim at maximizing the reliability of backup paths [37, 38]. Zheng *et al.* [89]

Table 5.11. The maximum logical link utilization when the average logical link utilization is 40%

Algorithm	ChinaNet	Level3	Qwest	XO
SelectBP-MP	100	100	100	100
Not-via	232.1	156.8	305.9	145.7
Not-via+SRLG	194.8	203.2	237.8	259.1
DR (0.2)	292.2	294.6	307.7	291.5
DR (0.5)	286.7	343.9	314.8	297.7
DR (0.8)	301.2	281.1	318.3	279.6

proposed an approach to on-demand search for the shortest recovery paths to recover intra-domain routing from large-scale failures. However, these approaches ignore the fact that a recovery path may not have enough bandwidth. Consequently, the rerouted traffic may cause severe logical link overload as observed on a major IP backbone by Iyer *et al.* [80].

There are some existing works that consider the traffic distribution among backup paths. Kvalbein *et al.* [90] proposed an approach to distribute the rerouted traffic without causing logical link overload. However, it considers that different backup paths have the same reliability, and it only uses the IP layer information for backup path selection. The approaches in [41, 42] address the logical link overload caused by backup paths. However, they aim at minimizing the bandwidth allocated to backup paths and consider logical link failures as independent events. The approach in [43] jointly addresses the recovery and traffic engineering in multi-path routing. Like R3, it aims at minimizing the maximum link utilization. Moreover, these two solutions do not consider the reliability of the backup paths.

All aforementioned approaches are built on single-layer information, whereas our approach is based on a cross-layer design. Another key difference is that our method jointly considers minimizing traffic disruption due to failures and avoiding logical link overload caused by the rerouted traffic.

Correlation between the logical and physical topologies: Some works on IP-over-WDM networks consider the correlation between the logical and physical topologies. Most of them focus on the survivable routing problem in IP-over-WDM networks [77, 82, 91, 92], i.e., building the mapping between logical and fiber links to minimize the impact of fiber link failures on logical links. Lee *et al.* [83, 84] analyzed the reliability of IP-over-WDM networks under fiber link failures.

They showed that the reliability of IP layer is strongly affected by the correlation between the logical and physical topologies. However, these works do not address the problem of selecting backup paths to protect IP links. Moreover, they do not model the impact of a logical link failure on the failure probability of fiber links, logical links, and backup paths, which is solved by our CFP model. Chigan *et al.* [93] protected IP routers by reusing optical wavelength channels. It deals with the optical layer configuration but does not consider the backup path selection in the IP layer. Furthermore, it only considers single failure; while our method does not have this assumption. Cui *et al.* [94] considered correlated failures in backup path allocation for overlay networks. However, they only use overlay layer information, while our approach is based on a cross-layer design. Moreover, they aim at finding reliable backup paths, whereas our objective is to minimize routing disruption.

Multi-path routing: Quality-of-Service (QoS) routing protocols [95, 96, 97, 98] use multiple paths between a pair of source and destination to achieve traffic engineering goals, e.g., minimizing the maximal link utilization. However, they do not consider the correlation between logical link failures. There are few recovery approaches that are built on multiple recovery paths. The approach in [41] aims at minimizing the bandwidth reserved for backup paths. It only uses IP layer information for backup path selection and assumes that the network merely has single logical link failure. R3 [40] reroutes traffic with multiple paths and the method in [43] jointly addresses failure recovery and traffic engineering in multi-path routing. They focus on traffic engineering goals rather than minimizing routing disruption. Moreover, they ignore the correlation between logical link failures and consider backup paths to have the same reliability.

5.6 Conclusions

The commonly used independent and SRLG models ignore the correlation between the optical and IP layer topologies. As a result, they do not accurately reflect the correlation between logical link failures and may not select reliable backup paths. We propose two cross-layer approaches for IP link protection. First, based on the topology mapping and failure probability of fiber links, we develop a correlated

failure probability (CFP) model to quantify the impact of IP link failure on the reliability of backup paths. With the CFP model, we propose a heuristic algorithm to choose the backup paths with minimum failure probability. We also propose a multi-round algorithm that considers the bandwidth constraint in backup path selection. It aims at choosing backup paths to minimize the traffic disruption caused by failures. Second, motivated by the discovery of the recent Internet measurements, we develop a probabilistically correlated failure (PCF) model which considers both independent and correlated IP link failures. The PCF model quantifies the impact of an IP link failure on the reliability of backup paths. With the PCF model, we propose an algorithm to minimize the routing disruption by choosing multiple reliable backup paths to protect each IP link. The proposed approach ensures that the rerouted traffic does not cause logical link overload, even when multiple logical links fail simultaneously. We evaluate the two cross-layer approaches using real ISP networks with both optical and IP layer topologies. Compared with prior works, the backup paths selected by our methods are much more reliable. Moreover, our approaches achieve higher recovery rate without causing logical link overload.

Chapter 6

Conclusions and Future Work

6.1 Summary

In this dissertation, we provide comprehensive solutions to detect and recover large-scale failures in the Internet. We summarize these techniques as follows.

In Chapter 2, we addressed two fundamental problems in probe-based network link monitoring, i.e., minimizing probing cost and achieving identifiability. Given a set of links to monitor, our objective is to select the minimum number of probing paths that can uniquely determine all identifiable links and cover all unidentifiable links. Based on linear system model, we proposed an algorithm to compute all irreducible sets of probing path which can uniquely determine an identifiable link. Then, we developed an extended bipartite model to reflect the relationship between a set of probing paths and an identifiable link. Finally, we proposed a heuristic-based algorithm to solve our problem. Simulations based on real network topologies showed that our approach can achieve identifiability with very low probing cost. Compared with prior work, our method is more general and has better performance.

In Chapter 3, we studied detecting and localizing large-scale router failures. To minimize the probing cost, our approach consists of a periodic detection phase and a localization phase that is triggered on demand. For the detection phase, we proposed a greedy algorithm to choose probing paths, so that we can probe all routers with minimal number of probing messages. For the localization phase, we discovered three types of probes that do not provide useful information and

avoided using them. We also developed a distance-based model to estimate the failure probability of the routers which cannot be reached by probes. Experimental results based on ISP topologies show that the accuracy of our approach is higher than 96.5%, even when only 10% of routers are connected by end systems for probing. Compared with prior works, the proposed approach achieves much higher accuracy with lower probing cost.

In Chapter 4, we presented an approach called Reactive Two-phase Rerouting (RTR) for intra-domain routing to quickly recover from geographically correlated failures with the shortest recovery paths. The basic idea is to collect information on failures, and then compute valid shortest paths and forward packets over them with source routing. To collect failure information, we designed a protocol to forward packets around the failure area and record failure information in the packet header. This process allows us to fully characterize the failed region quickly. Then, the second phase of RTR is invoked to calculate valid shortest paths to bypass the failure area and forward packets along the new shortest paths. RTR can deal with geographically correlated failures associated with areas of any shape and location, and is free of permanent loops. Compared with prior works, RTR achieves better performance for recoverable failed routing paths and uses much less network resources for irrecoverable failed routing paths.

In Chapter 5, we proposed two cross-layer approaches for intro-domain routing to recover from correlated link failures. Existing approaches have two limitations. First, the commonly used independent and Shared Risk Link Group models do not accurately reflect the correlation between IP link failures. Second, they ignore the traffic load and bandwidth constraint of IP links. Our solutions address both problems. The first approach protects each IP link with single backup path. We developed a correlated failure probability (CFP) model to reflect the correlation between IP link failures. With the CFP model, we proposed two algorithms for selecting a backup path to protect each IP link. The first algorithm focuses on maximizing backup path reliability, and the second algorithm aims at minimizing the traffic disruption caused by failures. Our second approach protects each IP link with multiple backup paths. We developed a probabilistically correlated failure (PCF) model based on the discovery of the recent Internet measurements. The PCF model considers both independent and correlated IP link failures. With

the PCF model, we proposed an algorithm to choose multiple reliable backup paths to protect each IP link. We evaluated the proposed approaches using real ISP networks with both optical and IP layer topologies. Compared with existing approaches, the backup paths selected by our approaches are much more reliable. Moreover, they achieve higher recovery rate without causing link overload.

6.2 Future Directions

Our work to date has provided a series of solutions for detecting and recovering from large-scale failures in the Internet, but there are still many other issues worthy of in-depth investigation. Next I outline several interesting directions for future work that one could pursue.

- **Failure detection with incomplete and noisy information:** Similar to existing works, our link monitoring and failure detection approaches described in Chapter 2 and Chapter 3 assume that the network topology is available and accurate. However, it is difficult to obtain accurate topology information, because the Internet is very large and keeps growing. Internet Service Providers have topology information, but they do not like to release these data because of business reasons. There have been many research works on inferring the Internet topology [99, 100, 52, 59, 101, 102]. However, the inferred topology is still inaccurate. Moreover, our work does not consider that some probes may not obtain useful information or even obtain incorrect information. For example, reply-based probe may be ignored by intermediate routers [46] or blocked by firewalls [47]. In these cases, we may incorrectly infer that there is a failure on the probing path. Therefore, new techniques are required to deal with incomplete and noisy information.

- **Detecting mixed link and router failures:** The recovery approach presented in Chapter 4 can deal with large-scale failures with both link and router failures, but our failure detection approaches address link failures and router failures separately. Detecting mixed link and router failures is very difficult and has not been investigated. The major challenge is to determine whether a failure is caused by a link or a router. A probe only checks if a

path is connected. For a disconnected path, the probe cannot determine if the path has a failed link or router. For small-scale failures, we can probe a link and a router from multiple directions to determine its status. However, for large-scale failures, probes may not reach some links and routers as explained in Chapter 3. We developed a distance-based model to estimate the failure probability of such kind of routers. To deal with mixed link and router failures, a new model is required to estimate the failure probability of links and routers.

- **Inter-domain routing recovery:** In Chapter 4 and Chapter 5, we focus on link-state protocols for intra-domain routing. Large-scale failures also have serious impact on inter-domain routing [9, 7, 103, 104]. Prior works on inter-domain routing recovery only deal with sporadic and isolated failures. They either reduce the protocol convergence time [105, 106, 103] or provide multiple routing paths to improve Internet reliability [107, 108]. Both approaches cannot effectively deal with large-scale failures. First, the BGP convergence time cannot be reduced to a satisfactory level. Second, multiple paths between the source AS and the destination AS may fail simultaneously when large-scale failures occur. The major challenge in inter-domain routing recovery is that routers do not have the AS-level topology. Consequently, the widely used backup path-based approaches are not suitable for this problem.

Appendix A

Correctness Proof of the Algorithm SolutionCalculation in Section 2.3.3

Theorem 10. *The algorithm SolutionCalculation can find all solutions to a solvable variable in the linear system.*

Proof. We prove it by contradiction. For a solvable variable x_i, suppose it has a solution S_q^i which is not found by the algorithm. The missed solution S_q^i cannot be linearly expressed by base solution S_1^i and linear expressions in set L. Otherwise, the algorithm SolutionCalculation can find it with linear replacement. Since solutions to x_i are linear combinations of variables b_j, S_q^i is in the following form.

$$x_i = \sum_{j=1}^{n} d_{qj} b_j \tag{A.1}$$

The above equation together with base solution S_1^i and linear expressions in L form a linear system as shown in Eq. (A.2). The first equation is base solution S_1^i. Since it is computed from linear system $A_R\mathbf{x} = \mathbf{b}_R$, the value of x_i is a linear combination of b_js, where $1 \leq j \leq r$. The second equation is the same as Eq. (A.1). For the other $n - r$ equations, each of them is a linear expression in set L, which is shown in Eq. (2.6).

$$\begin{cases} \sum_{j=1}^{r} d_{1j} b_j - x_i = 0 \\ \sum_{j=1}^{n} d_{qj} b_j - x_i = 0 \\ \sum_{j=1}^{r} c_{1j} b_j + b_{r+1} = 0 \\ \qquad\vdots \\ \sum_{j=1}^{r} c_{n-r,j} b_j + b_n = 0 \end{cases} \tag{A.2}$$

In this linear system, $b_1, \cdots, b_n$ and x_i are unknown variables. Hence, the coefficient matrix is shown in Eq. (A.3).

$$\begin{array}{c} \begin{array}{ccccccc} b_1 & \cdots & b_r & b_{r+1} & \cdots & b_n & x_i \end{array} \\ \begin{pmatrix} d_{11} & \cdots & d_{1r} & 0 & \cdots & 0 & -1 \\ d_{q1} & \cdots & d_{qr} & d_{q,r+1} & \cdots & d_{qn} & -1 \\ c_{11} & \cdots & c_{1r} & 1 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ c_{n-r,1} & \cdots & c_{n-r,r} & 0 & \cdots & 1 & 0 \end{pmatrix} \end{array} \tag{A.3}$$

we subtract $d_{q,r+k}$ times of the $(2+k)$th row from the second row, where $k = 1, \cdots, n-r$. For example, the second row is subtracted by $d_{q,r+1}$ times of the third row, $d_{q,r+2}$ times of the fourth row, and so on. The matrix turns into the form as shown in Eq. (A.4). Compared with the matrix in Eq. (A.3), the difference is only at the second row. For $j = 1, \cdots, r$, variable d'_{qj} is shown in Eq. (A.5).

$$\begin{array}{c} \begin{array}{ccccccc} b_1 & \cdots & b_r & b_{r+1} & \cdots & b_n & x_i \end{array} \\ \begin{pmatrix} d_{11} & \cdots & d_{1r} & 0 & \cdots & 0 & -1 \\ d'_{q1} & \cdots & d'_{qr} & 0 & \cdots & 0 & -1 \\ c_{11} & \cdots & c_{1r} & 1 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ c_{n-r,1} & \cdots & c_{n-r,r} & 0 & \cdots & 1 & 0 \end{pmatrix} \end{array} \tag{A.4}$$

$$d'_{qj} = d_{qj} - \sum_{k=1}^{n-r} c_{kj} d_{q,r+k} \tag{A.5}$$

Since row vectors in matrix A_R are linear independent, linear system $A_R\mathbf{x} = \mathbf{b}_R$ has only one solution to solvable variable x_i, i.e., base solution S_1^i. Therefore, the first two row vectors in Eq. (A.4) must be the same. Otherwise, there are two different linear combinations of $b_1, \cdots, b_n$ to express x_i, i.e., $A_R\mathbf{x} = \mathbf{b}_R$ has two different solutions to x_i. It shows that the second row vector of Eq. (A.3) can be linearly expressed by other row vectors. This contradicts with the assumption that S_q^i cannot be linearly expressed by S_1^i and linear expressions in set L. Therefore, no such missed solution S_q^i exists, which indicates that SolutionCalculation can find out all solutions to each solvable variable. □

Appendix B

Correctness Proof of the Algorithm PathSelection in Section 2.3.5

Theorem 11. *The set of probing paths returned by the algorithm PathSelection can uniquely determine all identifiable target links and cover all unidentifiable target links.*

Proof. Before removing redundancy, probing paths in set P_s can uniquely determine all identifiable target links and cover all unidentifiable target links. Thus, we need to prove that removing replaceable probing paths from P_s does not affect the correctness.

Suppose set P_s contains s probing paths $p_1, \cdots, p_s$, in which $p_{r+1}, \cdots, p_s$ are replaceable and removed from set P_s. Consider a removed probing path p_k. In the dependency matrix, row vector $\mathbf{a}_k$ can be linearly expressed by row vectors $\mathbf{a}_1, \cdots, \mathbf{a}_r$. Accordingly, variable b_k is a linear combination of $b_1, \cdots, b_r$ as shown in Eq. (B.1).

$$b_k = \sum_{i=1}^{r} c_i b_i \tag{B.1}$$

We first prove that removing redundancy does not affect uniquely determining identifiable target links. Given an identifiable target link e_j, it can be uniquely determined by probing paths $p_1, \cdots, p_s$. Hence, the linear system variable x_j can

be linearly expressed by variables $b_1, \cdots, b_s$ as shown in Eq. (B.2).

$$x_j = \sum_{i=1}^{s} d_i b_i = \sum_{i=1}^{k-1} d_i b_i + d_k b_k + \sum_{i=k+1}^{s} d_i b_i \tag{B.2}$$

Replacing b_k in Eq. (B.2) with Eq. (B.1) results in a new linear expression in Eq. (B.3).

$$x_j = \sum_{i=1}^{k-1} d_i b_i + d_k \sum_{i=1}^{r} c_i b_i + \sum_{i=k+1}^{s} d_i b_i \tag{B.3}$$

Since $r + 1 \leq k \leq s$, the above equation is equivalent to Eq. (B.4). It shows that x_j is a linear combination of $b_1, \cdots, b_{k-1}, b_{k+1}, \cdots, b_s$, which means that link e_k can be uniquely determined after removing the probing path p_k.

$$\begin{aligned} x_j &= \sum_{i=1}^{r} d_i b_i + \sum_{i=r+1}^{k-1} d_i b_i + d_k \sum_{i=1}^{r} c_i b_i + \sum_{i=k+1}^{s} d_i b_i \\ &= \sum_{i=1}^{r} (d_i + d_k c_i) b_i + \sum_{i=r+1}^{k-1} d_i b_i + \sum_{i=k+1}^{s} d_i b_i \end{aligned} \tag{B.4}$$

When removing probing path p_k, variable b_k in Eq. (B.3) is replaced by a linear combination of $b_1, \cdots, b_r$. We can remove all replaceable probing paths from set P_s with the same method. Finally, x_j becomes a linear combination of $b_1, \cdots, b_r$. It shows that e_j can be uniquely determined by $p_1, \cdots, p_r$.

Next, we prove that removing p_k does not affect covering unidentifiable target links. Suppose unidentifiable target link e_u is covered by p_k. Then, element $a_{k,u}$ in the dependency matrix is 1. Since row vector $\mathbf{a}_k$ can be linearly expressed by row vectors $\mathbf{a}_1, \cdots, \mathbf{a}_r$, elements $a_{1,u}, \cdots, a_{r,u}$ cannot be all 0, which means that at least one of $a_{1,u}, \cdots, a_{r,u}$ is 1. That is, at least one probing path among $p_1, \cdots, p_r$ can cover link e_u. Therefore, removing p_k does not affect covering e_u.

In conclusion, after removing redundancy, probing paths in set P_s can uniquely determine all identifiable target links and cover all unidentifiable target links. □

Bibliography

[1] Markopoulou, A., G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot (2008) "Characterization of Failures in an Operational IP Backbone Network," *IEEE/ACM Transactions on Networking*, **16**(4).

[2] Turner, D., K. Levchenko, A. C. Snoeren, and S. Savage (2010) "California Fault Lines: Understanding the Causes and Impact of Network Failures," in *ACM SIGCOMM*.

[3] Markopoulou, A., G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot (2004) "Characterization of failures in an IP backbone," in *IEEE INFOCOM*.

[4] US Department of Transportation (2001), "Effects of catastrophic events on transportation system management and operations: Baltimore, MD–Howard street tunnel fire," `http://www.itsdocs.fhwa.dot.gov/jpodocs/repts_te/13754_files/13754.pdf`.

[5] Cowie, J., A. Popescu, and T. Underwood (2005), "Impact of Hurricane Katrina on Internet infrastructure," `http://www.renesys.com/tech/presentations/pdf/Renesys-Katrina-Report-9sep2005.pdf`.

[6] LaPerriere, S. (2007), "Taiwan earthquake fiber cuts: a service provider view," `http://www.nanog.org/meetings/nanog39/presentations/laperriere.pdf`.

[7] Wu, J., Y. Zhang, Z. M. Mao, and K. G. Shin (2007) "Internet routing resilience to failures analysis and implications," in *ACM CoNEXT*.

[8] Ran, Y. (2011) "Considerations and Suggestions on Improvement of Communication Network Disaster Countermeasures after the Wenchuan Earthquake," *IEEE Communications Magazine*, **49**(1), pp. 44–47.

[9] Ogielski, A. and J. Cowie (2002), "Internet Routing Behavior on 9/11 and in the following weeks," `http://www.renesys.com/tech/presentations/pdf/renesys-030502-NRC-911.pdf`.

[10] Bellovin, S. M. and E. R. Gansner (2003) *Using link cuts to attack Internet routing*, *Tech. rep.*, AT&T Labs Research.

[11] Wilson, C. (2004), "High altitude electromagnetic pulse (HEMP) and high power microwave (HPM) devices: threat assessments," `http://www.fas.org/man/crs/RL32544.pdf`.

[12] Neumayer, S., G. Zussman, R. Cohen, and E. Modiano (2009) "Assessing the Vulnerability of the Fiber Infrastructure to Disasters," in *IEEE INFOCOM*.

[13] Neumayer, S. and E. Modiano (2010) "Network Reliability With Geographically Correlated Failures," in *IEEE INFOCOM*.

[14] Coltun, R., D. Ferguson, and J. Moy (1999), "OSPF for IPv6," RFC 2740.

[15] Hopps, C. (2008), "Routing IPv6 with IS-IS," RFC 5308.

[16] Rekhter, Y., T. Li, and S. Hares (2006), "A Border Gateway Protocol 4 (BGP-4)," RFC 4271.

[17] Iannaccone, G., C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot (2002) "Analysis of link failures in an IP backbone," in *ACM SIGCOMM Workshop on Internet measurment*.

[18] Giroire, F., A. Nucci, N.Taft, and C. Diot (2003) "Increasing the Robustness of IP Backbones in the Absence of Optical Level Protection," in *IEEE INFOCOM*.

[19] Shavitt, Y., X. Sun, A. Wool, and B. Yener (2001) "Computing the unmeasured: an algebraic approach to Internet mapping," in *IEEE INFOCOM*.

[20] Kompellay, R. R., J. Yatesz, A. Greenbergz, and A. C. Snoeren (2005) "IP fault localization via risk modeling," in *USENIX NSDI*.

[21] Li, F. and M. Thottan (2006) "End-to-end service quality measurement using source-routed probes," in *IEEE INFOCOM*.

[22] Zhao, Y., Y. Chen, and D. Bindel (2006) "Towards Unbiased End-to-End Network Diagnosis," in *ACM SIGCOMM*.

[23] Dhamdhere, A., R. Teixeira, C. Dovrolis, and C. Diot (2007) "NetDiagnoser: troubleshooting network unreachabilities using end-to-end probes and routing data," in *ACM CoNEXT*.

[24] Kompella, R. R., J. Yates, A. Greenberg, and A. C. Snoeren (2007) "Detection and localization of network black holes," in *IEEE INFOCOM*.

[25] Lee, P. P. C., V. Misra, and D. Rubenstein (2007) "Toward Optimal Network Fault Correction via End-to-End Inference," in *IEEE INFOCOM*.

[26] Nguyen, H. X. and P. Thiran (2007) "Network Loss Inference with Second Order Statistics of End-to-End Flows," in *ACM SIGCOMM Conference on Internet Measurement*.

[27] Zhao, Y., Z. Zhu, Y. Chen, D. Pei, and J. Wang (2009) "Towards Efficient Large-Scale VPN Monitoring and Diagnosis under Operational Constraints," in *IEEE INFOCOM*.

[28] Barford, P., N. Duffield, A. Ron, and J. Sommers (2009) "Network performance anomaly detection and localization," in *IEEE INFOCOM*.

[29] Cheng, L., X. Qiu, L. Meng, Y. Qiao, and R. Boutaba (2010) "Efficient Active Probing for Fault Diagnosis in Large Scale and Noisy Networks," in *IEEE INFOCOM*.

[30] Ghita, D., H. Nguyen, M. Kurant, K. Argyraki, and P. Thiran (2010) "Netscope: Practical Network Loss Tomography," in *IEEE INFOCOM*.

[31] Gu, Y., G. Jiang, V. Singh, and Y. Zhang (2010) "Optimal Probing for Unicast Network Delay Tomography," in *IEEE INFOCOM*.

[32] Kvalbein, A., A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne (2006) "Fast IP Network Recovery using Multiple Routing Configurations," in *IEEE INFOCOM*.

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

[34] Nelakuditi, S., S. Lee, Y. Yu, Z.-L. Zhang, and C.-N. Chuah (2007) "Fast local rerouting for handling transient link failures," *IEEE/ACM Transactions on Networking*, **15**(2).

[35] Zlatokrilov, H. and H. Levy (2008) "Area Avoidance Routing in Distance-Vector Networks," in *IEEE INFOCOM*.

[36] Kini, S., S. Ramasubramanian, A. Kvalbein, and A. F. Hansen (2009) "Fast Recovery from Dual Link Failures in IP Networks," in *IEEE INFOCOM*.

[37] Lee, H.-W. and E. Modiano (2009) "Diverse Routing in Networks with Probabilistic Failures," in *IEEE INFOCOM*.

[38] Hou, M., D. Wang, M. Xu, and J. Yang (2009) "Selective Protection: A Cost-Efficient Backup Scheme for Link State Routing," in *IEEE ICDCS*.

[39] Shand, M. and S. Bryant (2010), "IP Fast Reroute Framework," RFC 5714.

[40] Wang, Y., H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qiu, and Y. R. Yang (2010) "R3: Resilient Routing Reconfiguration," in *ACM SIGCOMM*.

[41] Banner, R. and A. Orda (2010) "Designing Low-Capacity Backup Networks for Fast Restoration," in *IEEE INFOCOM*.

[42] Johnston, M., H.-W. Lee, and E. Modiano (2011) "A Robust Optimization Approach to Backup Network Design with Random Failures," in *IEEE INFOCOM*.

[43] Suchara, M., D. Xu, R. Doverspike, D. Johnson, and J. Rexford (2011) "Network Architecture for Joint Failure Recovery and Traffic Engineering," in *ACM SIGMETRICS*.

[44] Stallings, W. (1998) *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 (3rd Edition)*, Addison-Wesley Longman Publishing Co., Inc.

[45] Song, H. H., L. Qiu, and Y. Zhang (2006) "NetQuest: a flexible framework for large-scale network measurement," in *ACM SIGMETRICS*.

[46] Gunes, M. H. and K. Sarac (2009) "Analyzing Router Responsiveness to Active Measurement Probes," in *Passive and Active Measurement Conference*.

[47] Sherwood, R. and N. Spring (2006) "Touring the Internet in a TCP sidecar," in *ACM SIGCOMM Conference on Internet Measurement*.

[48] Bejerano, Y. and R. Rastogi (2003) "Robust monitoring of link delays and faults in IP networks," in *IEEE INFOCOM*.

[49] Nguyen, H. X. and P. Thiran (2004) "Active Measurement for Multiple Link Failures Diagnosis in IP Networks," in *Passive and Active Measurement Conference*.

[50] Chen, Y., D. Bindel, H. Song, and R. H. Katz (2004) "An algebraic approach to practical and scalable overlay network monitoring," in *ACM SIGCOMM*.

[51] Agrawal, S., K. Naidu, and R. Rastogi (2007) "Diagnosing Link-level Anomalies Using Passive Probes," in *IEEE INFOCOM*.

[52] Bu, T., N. Duffield, F. L. Presti, and D. Towsley (2002) "Network tomography on general topologies," in *ACM SIGMETRICS*.

[53] Breitbart, Y., C.-Y. Chan, M. Garofalakis, R. Rastogi, and A. Silberschatz (2001) "Efficiently monitoring bandwidth and latency in IP networks," in *IEEE INFOCOM*.

[54] Li, L., M. Thottan, B. Yao, and S. Paul (2003) "Distributed network monitoring with bounded link utilization in IP networks," in *IEEE INFOCOM*.

[55] Thottan, M., L. E. Li, B. Yao, V. S. Mirrokni, and S. Paul (2004) "Distributed network monitoring for evolving IP Networks," in *IEEE ICDCS*.

[56] Suh, K., Y. Guo, J. Kurose, and D. Towsley (2005) "Locating network monitors: complexity, heuristics, and coverage," in *IEEE INFOCOM*.

[57] Zheng, Q., G. Cao, T. L. Porta, and A. Swami (2011) "Detecting and Localizing Large-Scale Router Failures Using Active Probes," in *IEEE MILCOM*.

[58] Golub, G. H. and C. F. V. Loan (1996) *Matrix Computations (3rd Edition)*, Johns Hopkins University Press.

[59] Sherwood, R., A. Bender, and N. Spring (2002) "Measuring ISP topologies with Rocketfuel," in *ACM SIGCOMM*.

[60] Nguyen, H. X., R. Teixeira, P. Thiran, and C. Diot (2009) "Minimizing Probing Cost for Detecting Interface Failures: Algorithms and Scalability Analysis," in *IEEE INFOCOM*.

[61] Zhang, Y., Z. M. Mao, and M. Zhang (2008) "Effective diagnosis of routing disruptions from end systems," in *USENIX NSDI*.

[62] Haeberlen, A., I. Avramopoulos, J. Rexford, and P. Druschel (2009) "NetReview: Detecting when interdomain routing goes wrong," in *USENIX NSDI*.

[63] Zheng, Q. and G. Cao (2010) "Minimizing Probing Cost and Achieving Identifiability in Network Link Monitoring," in *IEEE ICDCS*.

[64] Zhang, Y., Z. M. Mao, and J. Wang (2007) "A framework for measuring and predicting impact of routing changes," in *IEEE INFOCOM*.

[65] Francois, P., C. Filsfils, J. Evans, and O. Bonaventure (2005) "Achieving sub-second IGP convergence in large IP networks," *ACM SIGCOMM Computer Communication Review*, **35**(3).

[66] Basu, A. and J. Riecke (2001) "Stability issues in OSPF routing," in *ACM SIGCOMM*.

[67] Fang, Q., J. Gao, and L. J. Guibas (2004) "Locating and bypassing routing holes in sensor networks," in *IEEE INFOCOM*.

[68] Karp, B. and H. T. Kung (2000) "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *ACM MOBICOM*.

[69] Narvaez, P., K.-Y. Siu, and H. yi Tzeng (2000) "New Dynamic Algorithms for Shortest Path Tree Computation," *IEEE/ACM Transactions on Networking*, **8**(6).

[70] Alaettinoglu, C., V. Jacobson, and H. Yu (2000), "Towards Millisecond IGP Convergence," IETF Internet Draft.

[71] Gummadi, K. P., H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall (2004) "Improving the reliability of Internet paths with one-hop source routing," in *USENIX OSDI*.

[72] Nguyen, G. T. K., R. Agarwal, J. Liu, M. Caesar, P. B. Godfrey, and S. Shenker (2011) "Slick Packets," in *ACM SIGMETRICS*.

[73] Papagiannaki, K., S. Moon, C. Fraleigh, P. Thiran, and C. Diot (2003) "Measurement and Analysis of Single-Hop Delay on an IP Backbone Network," *IEEE Journal on Selected Areas in Communications*, **21**(6).

[74] Kim, Y.-J., R. Govindan, B. Karp, and S. Shenker (2005) "Geographic Routing Made Practical," in *USENIX NSDI*.

[75] Zheng, Q. and G. Cao (to appear) "Minimizing Probing Cost and Achieving Identifiability in Probe Based Network Link Monitoring," *IEEE Transactions on Computers*.

[76] Sharma, V. and F. Hellstrand (2003), "Framework for MPLS-based Recovery," RFC 3469.

[77] Crochat, O. and J.-Y. L. Boudec (1998) "Design Protection for WDM Optical Networks," *IEEE Journal on Selected Areas in Communications*, **16**(7).

[78] Oki, E., N. Matsuura, K. Shiomoto, and N. Yamanaka (2002) "A Disjoint Path Selection Scheme With Shared Risk Link Groups in GMPLS Networks," *IEEE Communications Letters*, **6**(9).

[79] Shen, L., X. Yang, and B. Ramamurthy (2005) "Shared Risk Link Group (SRLG)-Diverse Path Provisioning Under Hybrid Service Level Agreements in Wavelength-Routed Optical Mesh Networks," *IEEE/ACM Transactions on Networking*, **13**(4).

[80] Iyer, S., S. Bhattacharyya, N. Taft, and C. Diot (2003) "An approach to alleviate link overload as observed on an IP backbone," in *IEEE INFOCOM*.

[81] Rosen, E., A. Viswanathan, and R. Callon (2001), "Multiprotocol Label Switching Architecture," RFC 3031.

[82] Modiano, E. and A. Narula-Tam (2002) "Survivable Lightpath Routing: A New Approach to the Design of WDM-Based Networks," *IEEE Journal on Selected Areas in Communications*, **20**(4).

[83] Lee, K. and E. Modiano (2009) "Cross-Layer Survivability in WDM-Based Networks," in *IEEE INFOCOM*.

[84] Lee, K., H.-W. Lee, and E. Modiano (2010) "Reliability in Layered Networks with Random Link Failures," in *IEEE INFOCOM*.

[85] Kandula, S., D. Katabi, B. Davie, and A. Charny (2005) "Walking the Tightrope: Responsive Yet Stable Traffic Engineering," in *ACM SIGCOMM*.

[86] Guichard, J., F. le Faucheur, and J. P. Vasseur (2005) *Definitive MPLS network design*, Cisco Press.

[87] Fortz, B. and M. Thorup (2000) "Internet Traffic Engineering by Optimizing OSPF Weights," in *IEEE INFOCOM*.

[88] Bryant, S., M. Shand, and S. Previdi (2011), "IP fast reroute using Not-via addresses," IETF Internet Draft.

[89] Zheng, Q., G. Cao, T. L. Porta, and A. Swami (2012) "Optimal Recovery from Large-Scale Failures in IP Networks," in *IEEE ICDCS*.

[90] Kvalbein, A., T. Cicic, and S. Gjessing (2007) "Post-failure routing performance with multiple routing configurations," in *IEEE INFOCOM*.

[91] Kurant, M. and P. Thiran (2005) "On Survivable Routing of Mesh Topologies in IP-over-WDM Networks," in *IEEE INFOCOM*.

[92] Thulasiraman, K., M. S. Javed, and G. L. Xue (2009) "Circuits/Cutsets Duality and a Unified Algorithmic Framework for Survivable Logical Topology Design in IP-over-WDM Optical Networks," in *IEEE INFOCOM*.

[93] Chigan, C., G. W. Atkinson, and R. Nagarajan (2003) "Cost Effectiveness of Joint Multilayer Protection in Packet-Over-Optical Networks," *Journal of Lightwave Technology*, **21**(11).

[94] Cui, W., I. Stoica, and R. H. Katz (2002) "Backup Path Allocation Based On A Correlated Link Failure Probability Model In Overlay Networks," in *IEEE ICNP*.

[95] Orda, A. and A. Sprintson (2004) "Efficient Algorithms for Computing Disjoint QoS Paths," in *IEEE INFOCOM*.

[96] Ohara, Y., S. Imahori, and R. V. Meter (2009) "MARA: Maximum Alternative Routing Algorithm," in *IEEE INFOCOM*.

[97] Misra, S., G. Xue, and D. Yang (2009) "Polynomial time approximations for multi-path routing with bandwidth and delay constraints," in *IEEE INFOCOM*.

[98] Zhang, W., JianTang, C. Wang, and S. de Soysa (2010) "Reliable Adaptive Multipath Provisioning with Bandwidth and Differential Delay Constraints," in *IEEE INFOCOM*.

[99] Burch, H. and B. Cheswick (1999) "Mapping the Internet," *IEEE Computer*, **32**.

[100] Govindan, R. and H. Tangmunarunkit (2000) "Heuristics for Internet map discovery," in *IEEE INFOCOM*.

[101] Ni, J., H. Xie, S. Tatikonda, and Y. R. Yang (2008) "Network routing topology inference from end-to-end measurements," in *IEEE INFOCOM*.

[102] Sherwood, R., A. Bender, and N. Spring (2008) "DisCarte: A disjunctive Internet cartographer," in *ACM SIGCOMM*.

[103] Sahoo, A., K. Kant, and P. Mohapatra (2006) "Improving BGP convergence delay for large-scale failures," in *IEEE/IFIP DSN*.

[104] Popescu, A., B. Premore, and E. Zmijewski (2008), "Impact of the Middle East Cable Breaks: A Global BGP Perspective," `http://www.renesys.com/tech/presentations/pdf/nanog42-lightning.pdf`.

[105] Bremler-Barr, A., Y. Afek, and S. Schwarz (2003) "Improved BGP convergence via ghost flushing," in *IEEE INFOCOM*.

[106] Bonaventure, O., C. Filsfils, and P. Francois (2005) "Achieving sub-50 milliseconds recovery upon BGP peering link failures," in *ACM CoNEXT*.

[107] Xu, W. and J. Rexford (2006) "MIRO: multi-path interdomain routing," in *ACM SIGCOMM*.

[108] Kushman, N., S. Kandula, D. Katabi, and B. Maggs (2007) "R-BGP: staying connected in a connected world," in *USENIX NSDI*.

Vita

Qiang Zheng

Qiang Zheng received the B.S. degree in Computer Science from Nankai University, Tianjin, China, in 2004, and the M.E. degree in Computer Science from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2007. He enrolled in the Ph.D. program in Computer Science and Engineering at The Pennsylvania State University in August 2007. He is a student member of IEEE.

Publications during the Ph.D. study:

- Qiang Zheng and Guohong Cao, "Minimizing Probing Cost and Achieving Identifiability in Probe Based Network Link Monitoring," IEEE Transactions on Computers, to appear.
- Qiang Zheng, Jing Zhao, and Guohong Cao, "A Cross-Layer Approach for IP Network Protection," IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2012.
- Qiang Zheng, Guohong Cao, Tom La Porta, and Ananthram Swami, "Optimal Recovery from Large-Scale Failures in IP Networks," IEEE International Conference on Distributed Computing Systems (ICDCS), 2012.
- Qiang Zheng, Guohong Cao, Tom La Porta, and Ananthram Swami, "Detecting and Localizing Large-Scale Router Failures Using Active Probes," IEEE Military Communication Conference (MILCOM), 2011.
- Qiang Zheng and Guohong Cao, "Minimizing Probing Cost and Achieving Identifiability in Network Link Monitoring," IEEE International Conference on Distributed Computing Systems (ICDCS), 2010.