Backup Route Design for Reliable and Sustainable Packet Networks

高信頼パケットネットワークのための予備経路設計法の研究

March 2013

Graduate School of Global Information and Telecommunication Studies
Waseda University

Distributed Computing System II

Shohei Kamamura

鎌村　星平

Abstract

The goal of this study is to improve reliability and sustainability of backbone network. Concretely, we study the carrier-grade IP fast rerouting, and a dynamic path protection for MPLS network. IP fast rerouting was developed for *the Internet*, and was expected to realize sub-50ms restoration. For applying carrier networks, however, requirements for IP fast rerouting is not only recovering within sub-50ms, but also having scalability and handling multiple failures. This paper mainly studies scalability issue in Chapter 2 and extension of restoration range in Chapter 3 respectively. In terms of implementation, IP fast rerouting requires the specific forwarding implementation for a router. This prevents the IP fast rerouting to becoming widespread. In Chapter 4, we also study the implementation issue, which utilizes OpenFlow and minimizes the impact to existing hardware. Finally, Chapter 5 studies a relaxation of maintenance time for MPLS path protection network.

Acknowledgments

This dissertation completes my achievements of research activities at Graduate School of Global Information and Telecommunication Studies, Waseda University, Japan, under the supervision of Prof. Hidenori Nakazato. I would like to express my gratitude to Prof. Nakazato for his guidance and encouragement.

I also would like to appreciate Prof. Toshitaka Tsuda, Prof. Yong-Jin Park and Prof. Yoshiaki Tanaka for their supervision for this dissertation. Prof. Tanaka indirectly coaches my studies through domestic and international academic conferences.

My work presented in this dissertation was also supported by NTT Network Service Systems Laboratories. I would like to appreciate my supervisors, Mr. Hedehiro Arimitsu, Dr. Koji Sasayama, Dr. Yoshihiko Uematsu, Mr. Takashi Miyamura, and Mr. Daisaku Shimazaki. I also would like to appreciate my former supervisors, Mr. Atsushi Hirmatsu and Dr. Kohei Shiomoto. They gave me an opportunity to get Ph.D while working for NTT Network Service System Laboratories.

I would like to express my sincere appreciation to my mother Kuniko for her support and encouragement that I received since my birth. I would not be where I am today without them. My brother Ippei also played an important role for my campus life when I was a scholarship student.

Finally, I would like to express my thanks and my love to my wife Mikako and my daughter Hinako. Their smiles give me energy and bravery.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Internet Protocol (IP) became a core technology for not only *the Internet* but also some of carrier networks such as next generation network (NGN) [1]. The most peculiar point of IP is adaptability to network environment changes. In IP network, alternative packet routes are autonomously provided by each IP router when network state changes because of periodical maintenance or failures [2]. That is, reachability among any routers is assured as long as physical network connectivity is maintained. On the other hand, this autonomous property requires the reactive actions after network changes: collecting the updated network topology information, recomputing the alternative routes, and updating the database in router, which stores the route information. These reactive actions should be also synchronized among all routers. The total time for the above-mentioned reaction, which also mean restoration time from a failure, becomes over several tens of seconds in a worst case. During this time, the packets which do not have alternative routes are dropped. Long restoration time is a barrier to achieving carrier-grade service level agreement (SLA).

As an alternative, multi-protocol label switching (MPLS) technology, which provides connection-oriented paths, was also proposed for a backbone network [3]. Recently, MPLS transport profile (MPLS-TP) [4], which extended operations, administration and management (OAM) functions for MPLS, was also proposed. These technologies enable network operators to establish explicit paths among nodes, and network is operated in accordance with the operator's proactive network design. MPLS provides fast restoration against a failure by 1+1 path protection mechanism. The 1+1 path protection provides primary and backup paths, which are link- or node-disjoint paths. Traffic data is transmitted on both paths, and the receiver node switches receiving data; therefore, it can achieve lossless data transmission even on the failure of the one of the paths. On the other hand, unlike IP, MPLS cannot flexibly handle unexpected network changes because paths are statically designed.

The goal of this study is to provide reliable and sustainable network design by enhancing both IP and MPLS respectively. Because each technology has different domain of applicability, enhancement of each technologies is required; upper layer

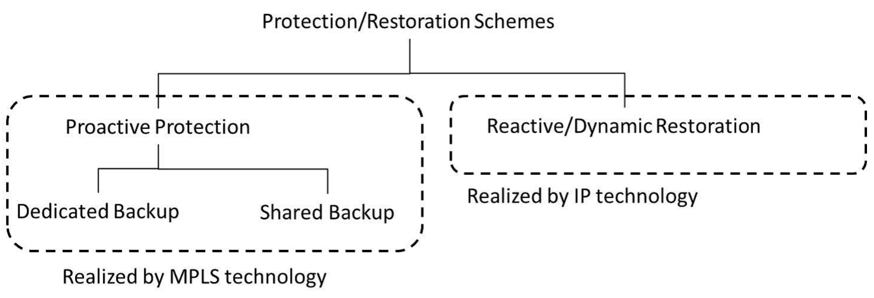

Protection/Restoration Schemes
Proactive Protection
Dedicated Backup
Shared Backup
Realized by MPLS technology
Reactive/Dynamic Restoration
Realized by IP technology

Figure 1.1: Taxonomy of reliable network design

IP restoration is generally superior to MPLS in terms of cost-effective restoration, and MPLS provides a potential for traffic engineering to network operators. To complement drawbacks of both technologies, we study an IP fast rerouting, which can also handle carrier-grade requirements, and a dynamic path protection for MPLS network.

In this chapter, we first describe taxonomy of existing reliable network designs in Section 1.2. Section 1.3 provides objectives of this dissertation. The outline of this dissertation is shown in Section 1.4

1.2 Taxonomy of Reliable Network Design

As shown in Fig. 1.1, technologies for reliable network design are classified roughly into 2 classes: proactive protection and dynamic restoration [5]. In the protection technology, backup routes are precomputed and preconfigured. Fast restoration against a failure is realized by skipping some reactive recovery actions. On the other hand, reactive restoration technology computes and configures backup routes after a failure. This means that though the restoration time increases compared to the protection technology, backup routes are optimized in accordance with the changed network topology. The proactive protection is realized by explicit path configuration with MPLS, and the dynamic restoration is realized by IP routing protocol such as open shortest path first (OSPF) [2] and/or border gateway protocol (BGP) [13]. OSPF is used for controlling single domain routing. In an OSPF network, each router has the whole network topology information, and it is synchronized among routers. A route from a source router to a destination router is computed based on a link metric, which is assigned to each link in the network. The shortest path, whose summation of link metrics is minimal, is provided as the route. BGP is used for exchanging route information between multiple network domains. BGP distributes routing information with some attributes, and it indirectly determines the intra-domain and/or inter-domain routes.

For the IP reactive restoration, a main issue in terms of improving IP resilience is its long restoration time. As shown in Fig. 1.2, there are some methods to shorten the restoration time. As reactive-approaches, the full flooding and limited flooding were proposed. The former is classified as the fast convergence and the efficient resource management. For the fast convergence, minimization of

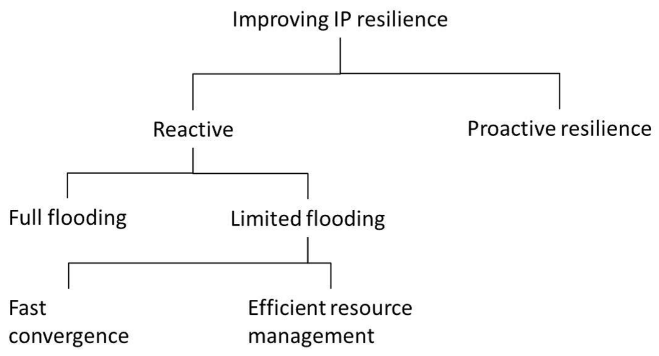


Figure 1.2: Improving IP resilience

the convergence time [6, 14, 15] and the computation time [7] for OSPF or BGP were proposed. For the efficient management, routes are preliminary optimized by assuming possible network changes [8–11]. For example, heuristic algorithms for the link metric design under the OSPF network to minimize the maximum link utilization was proposed [8, 10]. Wang et al. proposed the link metric optimization method using mathematical programming under unequal splitting of OSPF environment [9]. Fortz and Thorup also proposed the link metric design, which considered the operation overhead by limiting the number of metric changes [11]. In the limited flooding, convergence time is shortened by limiting notification area of OSPF flooding [12].

Recently, IP fast rerouting, which embeds the concept of MPLS proactive protection to IP, have been proposed. There are technologies for intra-domain control [24–37], and for inter-domains control [16–23]. The basic strategy of IP fast rerouting is precomputing backup routes like MPLS path protection, and switching primary routes to backup routes after a failure occurrence. In pure IP network, because there are no schemes to maintain the precomputed backup routes, IP fast rerouting technologies require extension of the current IP framework. For example, backup topology-based approach [24–29] has multiple routing tables in a router, and uses them in accordance with network status. Loop-free alternate approach [30] preliminary stores the alternative next hop information against failure occurrences, and not-via approach [31] indirectly has next hop information, which is realized by encapsulation of IP addresses. For other IP fast rerouting [32–37], though the mechanism of failure detection and packet forwarding are different, basic strategy is preparing backup tables, which store precomputed backup routes. For BGP networks, similar method is applied: storing precomputed backup routes [16–21], using precomputed tunnel [22], and having multiple control planes for BGP [23].

1.3 Objectives of This Dissertation

As shown in Fig. 1.3, IP and MPLS are used for realizing reactive restoration and proactive protection respectively in terms of reliable and sustainable packet net-

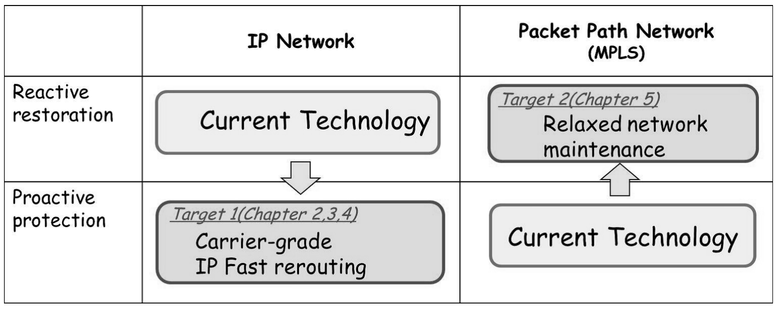


Figure 1.3: Objectives and overview of this dissertation

work design. As a result, the main disadvantage of IP is the longer restoration time though it can handle unexpected network changes by its autonomous and reactive property. Inversely, the main disadvantage of MPLS is low adaptability to network changes though its proactive restoration mechanism can achieve fast restoration. Though there were many studies for shortening IP restoration time based on IP reactive property, their restoration time did not reach the level of MPLS proactive protection. For example, the goal of [6,7,12,14,15] is minimizing convergence time: convergence time does not become zero. The goal of predesign approaches [8–11] is producing the total optimization solution by assuming multiple network changes: routes are not necessarily optimized for each network configuration. Against these technologies, IP fast rerouting [16–37] is based on the concept of path protection, which prepares the precomputed routes. It can provide short restoration time, which is equivalent to MPLS path protection. That is, IP fast rerouting is a typical case in which a technology is improved by deriving an idea from an alternative technology: we believe that this research acts as a catalyst for technical breakthrough.

Because IP and MPLS respectively have different domain of applicability, enhancement of each technologies is required. For example, IP restoration is generally superior to MPLS in terms of cost-effective restoration because it could handle both the under layer failures (e.g., fiber-cut) and upper layer failures (e.g., router interface down). On the other hand, MPLS has potential for traffic engineering by utilizing its explicit route design framework.

From the above observation, the goal of this study is to improve reliability and sustainability of backbone networks by enhancing both technologies: IP and MPLS. Concretely, we study the carrier-grade IP fast rerouting and the dynamic path protection for MPLS network (Fig. 1.3). Noted that this study focused on control of the intra domain network, which is operated by a single carrier. Study for BGP network is outside the scope of this paper.

Followings are issues covered in this paper.

Enhancement of IP Fast Rerouting

Firstly, we study the IP fast rerouting, which satisfies the carrier-grade requirements. Carrier-grade requirements for the failure restoration are as follows.

1. fast restoration (sub-50ms)

2. scalability for the large-scale network

3. handling multiple failures

Basically, IP fast rerouting was developed for *the Internet*. Then, requirements (2) and (3) were not specifically considered. Therefore, this paper mainly studies scalability issue and extension of restoration range.

Secondly, we also tackle the implementation issue of IP fast rerouting. As previously described, IP fast rerouting requires the specific forwarding mechanism. This prevents the IP fast rerouting to becoming widespread. In this paper, we challenge to realize the implementation method which minimizes the modification of existing hardware.

Enhancement of MPLS Path Protection

For enhancement of MPLS path protection network, we challenge to the problem for relaxing the maintenance time for 1+1 path protection environment.

1.4 Dissertation Outline

The remainder of the dissertation is organized as follows. Chapter 2 addresses the scalability improvement of IP fast rerouting. Noted that most part of Chapter 2 consists of my previous works before the doctoral program. However, it provides a key concept of IP fast rerouting and guidance of deep understanding for the following chapters. Therefore, in this chapter, we present our previous study in detail. Chapter 3 addresses the IP fast restoration problem from concurrent double failures. Chapter 4 presents an implementation of IP fast rerouting without hardware modification. Chapter 5 addresses relaxed maintenance for MPLS path protection network. Finally, Chapter 6 concludes the dissertation. Below we present an outline for each chapter.

1.4.1 IP Fast Rerouting Using Spanning Tree-Based Backup Topologies

Chapter 2 introduces the basic concept of IP fast rerouting using backup topology. In this framework, backup routes are precomputed based on the backup topologies. An arbitrary single failure is protected in at least one backup topology. When a packet encounters a failure, a backup topology ID is attached to the packet header. There are backup forwarding tables associated with backup topologies. Then, the packet is sent to the backup next hop designated by the backup forwarding table by referring to the backup topology ID. These precomputation-based forwarding mechanisms prune away the reactive actions such as global failure notification and recomputation of backup routes, and then fast restoration is realized.

In this chapter, we first introduce the existing works for IP fast rerouting. Then, we introduce our fundamental backup topology design algorithm, which minimizes the number of backup topologies. In IP fast rerouting framework using backup topologies, the number of forwarding tables is proportional to the number of backup topologies. In addition, length of available packet header field (e.g., ToS) is limited. Then, minimization of the number of backup topologies becomes an important issue. In our proposal, each backup topology is made such that the available links and nodes in each backup topology consists spanning tree: this strategy maximizes the number of protections in each backup topology, and then the number of backup topology is reduced. We showed that the number of backup topologies is reduced up to 60% compared to the existing design algorithm.

1.4.2 Loop-free IP Fast Rerouting considering Double Link Failures

Chapter 3 proposes an IP fast rerouting whose restoration range is extended from a single failure to double failures. In particular, we tackle the suppression of the forwarding loop while keeping the number of backup topologies.

We propose a novel forwarding algorithm which avoids forwarding loop while using only the backup topology ID contained in the packet header. To avoid the forwarding loop, failure detecting router only has to notify the explicit failed point. However, available IP header field is finite (e.g. 8 bit), and then applicability of this approach is restricted up to 80 nodes network. Our algorithm estimates the first failure point from the backup topology ID: this algorithm could reduce the occurrence of forwarding loops while it does not limit applicable network size for practical use.

We also propose a backup topology design algorithm considering double link failures while suppressing the number of backup topologies. When the restoration target is extended from single-link failures to double-link failures, the number of failure patterns becomes the combination of two-links. A sufficient condition for achieving loop-free restorations is that arbitrary two-links are protected on one of the backup topologies, and then the required number of backup topologies is large. In this chapter, we focus on the fact that the above condition is not necessary and causes redundant backup topologies because the backup route could be constructed by a combination of backup topologies. Concretely, our backup topology design algorithm increases the diversity of backup routes by raising the frequency of appearance of links, which have high betweenness centrality, on backup topologies. The betweenness centrality is a measure of appearance of a link in paths, and it equals to the number of shortest paths from all vertices to all others that pass through that link. We express the characteristic of a topology by betweenness centrality of links. We derive the set of backup topologies whose aggregated number of links is proportional to the betweenness centrality of the original topology. We express both the betweenness centrality in the original topology and the number of links connecting the same adjacent nodes in the network where backup topologies are superimposed as adjacency matrices. Then we calculate the principal eigenvectors of both matrices. In this way, their similarity can be easily observed by computing the cosine similarity of eigenvectors.

Our packet forwarding algorithm can reduce the average loop probability from

10^{-2} order to 10^{-3} order compared to the existing algorithm. For the backup topologies design, our algorithm can reduce the number of backup topologies about 35-50% compared to the simple benchmark algorithm, where arbitrary two-link pairs are protected on one of the backup topologies.

1.4.3 Implementation Design of IP Fast Rerouting using OpenFlow

Chapter 4 proposes an implementation design for IP fast rerouting using OpenFlow. As previously described, many algorithms for backup topology design were proposed. In addition, a framework, which constructs multiple forwarding tables in a router, was standardized by IETF. However, there were few discussions for implementation and there was no implementation for commercial routers. This is because that IP fast rerouting requires the coordination of specific forwarding functions to a forwarding hardware, and then its implementation becomes more complicated. In this chapter, we focus on the OpenFlow protocol, which physically separates control functions from forwarding hardware, and places them in software controllers. To store multiple forwarding tables, we utilize the pipeline processing with multiple flow tables defined in OpenFlow switch specification 1.1. Then, forwarding functions, which are specifically required for IP fast rerouting, are implemented in the programmable controller. That is, we could achieve IP fast rerouting without any extension of current forwarding hardware. On the contrary, increase of backup routes becomes main overhead of our proposal. We also embed the compression mechanism of backup routes using shared memory to our IP fast rerouting implementation design.

This chapter provides validity of our implementation design for IP fast rerouting through computer simulation. Firstly, we showed that our proposal could achieve sub-50ms restoration without any extension of current forwarding hardware. In addition, our flow table compression mechanism can reduce the size of flow table up to 50% compared to the existing algorithm.

1.4.4 Relaxed Maintenance Network using Dynamic 1+1 Path Protection

Chapter 5 presents the dynamic path reconfiguration method in 1+1 path protection network for relaxing the restriction in terms of the maintenance time. The 1+1 path protection provides fast restoration by preliminary establishing both primary and backup paths. In terms of network operation, when network equipment (physical link or node) failed, network operator should rapidly repair the failed equipment to satisfy the certain level of availability (e.g., 99.9999%). However, repairing physical failure such as fiber-cut requires long time which includes equipment procurement. In addition, failure occurrence at night needs expensive human resources. In this chapter, we propose a network architecture and path computation algorithm to maintain 1+1 path protection as much as possible after a single failure by dynamically assigning new backup path. Establishing new backup path in a few seconds realize both the minimization of the duration in which the target availability is not satisfied, and relaxing of the restriction in terms of the maintenance time.

In this chapter, we evaluated our proposal in two perspectives: network design problem and network sustainability problem. From the network design perspective, we evaluated total equipment cost, path length, and blocking probability of our proposed dynamic approach and a static design approach on various forms of networks. The static design approach we used is the one which establishes three independent paths in advance. The number of sparse-node becomes the main factor of an increase of blocking probability. Our dynamic path protection, which behaves like a link protection mechanism, could maximally avoided the influence of sparse-node existence, and then reduce blocking probability about 53% compared to the static design. From the network sustainability perspective, our algorithm, which utilizes available resources as much as possible, could reduce blocking probability about 10%-20% compared to simple benchmark algorithms. It also results in increase of allocatable maintenance time. Using our method, the portion of failures which allow more than 100-hour repair time increases 10%.

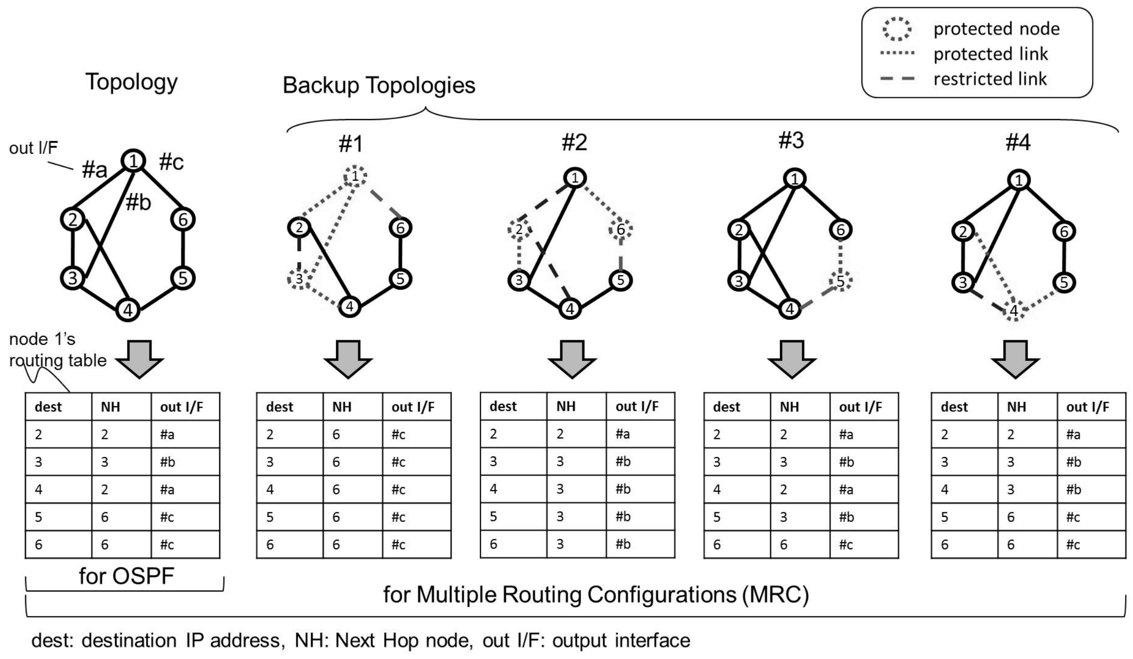
Chapter 2

IP Fast Rerouting Using Spanning Tree-Based Backup Topologies

2.1 Introduction

Link-state routing protocols, such as Open Shortest Path First (OSPF) [2], are being widely used for intra-domain routing resolution, but finding alternate routes requires a few seconds after a failure occurs [24]. This recovery time is too long to achieve robustness for the increasing number of multimedia applications. IP fast rerouting techniques have been studied to achieve recovery within just a few milliseconds [24–37]. The basic idea of IP fast rerouting is to reduce recovery time after a failure by precomputing backup routes. The multiple routing configurations (MRC) method has been proposed for IP Fast Rerouting [24]. The MRC method prepares backup topologies, which are precomputed and used for finding a detour route after a failure. In a backup topology, some links are assigned a higher metric value. A metric is integral value, and represents the cost of the link for path computation. Such links are called protected links and the backup topology provides detour routes on the failures of those links. Every link is required to be a protected link in at least one backup topology. In this way, we can achieve fast recovery against any single failure by using backup topologies.

We define the backup topology-creation problem tackled here as minimizing the number of backup topologies for ensuring network scalability and applicability to real networks. The total number of backup topologies is an important factor. Requiring too many backup topologies consumes a large share of router resources. This is because the sizes of the routing table kept in a router is proportional to the number of backup topologies. Moreover, if the required number of backup topologies exceeds the available number which is restricted by router configuration, fast recovery cannot take place for some failures because not-necessarily all backup topologies can be installed in every router. Therefore, minimization of the number of backup topologies is the main issue for achieving scalable and applicable IP fast rerouting.



Topology (for OSPF):

dest	NH	out I/F
2	2	#a
3	3	#b
4	2	#a
5	6	#c
6	6	#c

#1:

dest	NH	out I/F
2	6	#c
3	6	#c
4	6	#c
5	6	#c
6	6	#c

#2:

dest	NH	out I/F
2	2	#a
3	3	#b
4	3	#b
5	3	#b
6	3	#b

#3:

dest	NH	out I/F
2	2	#a
3	3	#b
4	2	#a
5	3	#b
6	6	#c

#4:

dest	NH	out I/F
2	2	#a
3	3	#b
4	3	#b
5	6	#c
6	6	#c

Figure 2.1: Overview of backup topologies

We propose a backup-topology-creation algorithm for minimizing the number of backup topologies. The key point of our algorithm is that each backup topology is made such that the topology, excluding the protected links, forms a spanning tree. A spanning tree of a given topology is a minimal set of links that connects all nodes. Therefore, this process maximizes the number of protected links in one backup topologies. The next sub-problem is how to generate the minimal set of spanning trees that protects all the links. We solve this sub-problem by link-weight manipulation considering link properties. The evaluation results show that our algorithm achieves fast restoration of all traffic with only a few backup topologies.

The rest of the chapter is organized as follows. In Section 2.2, we introduce the overview of IP fast rerouting, related works, and our problem statement. In Section 2.3, our new backup topology-creation algorithm is presented. Our evaluation results are shown in Section 2.4. Finally, we conclude our discussion in Section 2.5.

2.2 Overview of IP Fast Rerouting

In this section, we describe the characteristics of backup topologies used with the MRC method and introduce IP Fast Rerouting using backup topologies. We also provides related works, and then we state our problem.

2.2.1 Overview of Backup Topologies

The characteristics of backup topologies defined by Kvalbein et al. [24] are as follows (Fig. 2.1). A backup topology consists of normal links, protected links, restricted links, and normal and protected nodes. A protected link is a link that may fail and is protected by the backup topology, and a restricted link is a link that can be used only as the first or last hop for packets that use the backup topology. A link failure is protected by using protected link, and a node failure is protected by using protected node. The metric of a protected link is set to the maximum value provided by the link-state routing protocol, and the metric of a restricted link is set to a large value, although it is not the maximum value [24]. A protected node is a node that is only connected to protected and restricted links. That is, protected links and nodes are the resources protected by backup topologies. They are not used to forward traffic when a resource fails.

Backup topologies should satisfy the following characteristics for protecting not only any single link failure but any node failure.

1. Each backup topology is a connected graph that does not contain protected links.

2. The union of the protected nodes and links of all backup topologies corresponds to the original topology.

3. Among the links that are connected to a protected node, at least one is a restricted link, and the others are protected links. The opposing node of a restricted link must not be a protected node in the same backup topology.

If backup topologies satisfy the above conditions, an arbitrary link is a protected link in at least one backup topology. That is, any single link failure can be protected. In addition, backup topology can handle any single node failure. Handling a node failure means that it provides a detour route which does not use links which connect failed node except when the failed node is the destination node. Failure-detecting node detects a link failure even if a link failure is caused by a node failure. Then failure-detecting node selects backup topology whose node connected with failed link is protected node. In packet forwarding process by using above backup topology, the links that are connected with the failed node are not used except when the failed node is the destination node. This is because the failed node is the protected node: it is only connected to protected and restricted links. Therefore, the detour route determined by the backup topology avoids the protected node. Thus, for each single failure, there is a backup topology that avoids the failed resources.

2.2.2 IP Fast Reroute using Backup Topologies

IP fast rerouting can be achieved by using backup topologies [24]. Each backup topology is precomputed and installed in routers. Backup topologies are used to define different topologies, which employ different metrics. Each router computes the shortest path and then creates the routing entries (relationship between destination IP address and next hop node) based on the original topology, and for each backup topology. If the router detects a link failure, it searches for the backup topology that protects the failed resource. Next, the identifier of the selected backup topology is marked in the type of service (ToS) field of the IP header. After this marking, the failure-detecting router forwards the packets to the next hop node according to the routing entry of the selected backup topology. Other routers can forward the IP packets according to the same backup topology by referring to the ToS field. We explain the example using Fig. 2.1. If node 1 detects the link 1-3 failure when the packets whose destination addresses are node 3 arrive at node 1, node 1 selects backup topology #1 because the failed link (1-3) is protected. Then, these packets are forwarded to nodes 6, 5, 4, 2, and 3 according to backup topology #1 by referral to the ToS field. Protocol extensions called multi topology routing [39] is being standardized for realizing IP fast rerouting.

While the current IP restoration requires long time for restoration, it could provide globally optimized backup routes through global convergence. Then, the IP fast rerouting scheme can also work with the current IP restoration mechanism. In parallel with the recovery process of IP fast rerouting, updated link state advertisements (LSA) are flooded to each router, and new routes are computed. As previously described, this global convergence requires long time, and then packet forwarding based on the IP fast rerouting are performed until the current IP restoration is resumed. That is, IP fast rerouting can also play a role as supplement of the current IP restoration.

2.2.3 Related Works

Motivation of IP fast rerouting is realizing millisecond-order recover time without alternate packet forwarding techniques such as MPLS [3]. In this section, we introduce existing IP fast rerouting techniques [24–37], and qualitatively evaluate them.

Many algorithms for backup topology design have been proposed [24–29]: these involve basic design [24], minimizing the number of backup topologies [25, 26], load-balancing of backup routes [28, 29], and considering multiple failures [27]. In addition, a framework, which constructs multiple routing tables, is standardized by IETF [39].

A loop-free alternate (LFA) [30], which does not require extensions to existing link-state routing protocols to set the proper link metrics, has been proposed. The node that detects a failure forwards the IP packets to the node called the LFA, which is not the original next hop. The link metrics are set so that the route from the LFA to the destination node does not include the node detecting the failure. Therefore, routing loops are avoided. This technique was standardized by IETF. However, the drawback of this technique is lack of flexibility: it cannot handle single failures in arbitrary topologies because routes based on link metrics depend on the topology [38].

A not-via address approach has also been proposed [31]. The key idea of not-via address is tunneling: IP packets are encapsulated and a header is added to the packets. This header contains the not-via address. The encapsulated packets detour around failure points because the route for a not-via address is precomputed to avoid the failure points. This header with the not-via address is removed at nodes located downstream of the failure points, and the IP packets are forwarded according to the original destination address. Generally, the drawback of the tunneling approach may give less optimal backup paths, and less flexibility with regards to post failure load balancing.

Nelakuditi et al. [32] proposed an IP fast rerouting technique called failure insensitive routing (FIR). The key point of FIR is that a failure point is estimated from the relationship between the destination IP address of an incoming packet and incoming interface. If packets are received on an irregular interface, which is connected with the next hop node on the shortest path to the destination, the router interprets that as a failure having occurred on the shortest path. The advantage of this estimating approach is that there is no need to advertise the failure point. FIR is extended to handle node failure [33] and to work on an inter-domain environment [34]. However, it cannot handle multiple failures, and it needs extension of commercial routers.

Xi. et al proposed their IP fast rerouting called ESCAP [35]. Their algorithm define the bypass port called backup port, and it is used for rerouting. Their scheme and FIR [32] share similar ideas. The difference is that they develop a different algorithm that does not have any assumptions on the primary paths: the primary paths can be either shortest or non-shortest. They also proposed an IP fast rerouting scheme for IP multicast inspired by the ring protection mechanism [36], and an IP fast rerouting considering SRLG failure [37]. However, like FIR algorithm [32–34], it also requires their own extension.

Our attention in deploying the IP fast rerouting is its flexibility and applicability to actual networks. For the flexibility such as backup path optimization and recovering from various failures, MRC approach [24–29], FIR approach [32–34], and ESCAP [35–37] frameworks are good solutions. On the other hand, for the applicability to actual networks, standardization is a good indication, and MRC, LFA [30], and not-via address [31] frameworks are suitable. Therefore, we believe that MRC framework, which has both the flexibility and applicability, is promising technology for extending the current IP network to more resilient and reliable network.

2.2.4 Problem Statement

Minimizing the number of backup topologies [25, 26] is the most important issue for ensuring network scalability and applicability to actual condition.

For the network scalability, memory size of routers is the main issue. This is because the size of the routing table kept on a router is proportional to the number of backup topologies. An example is shown in Fig. 2.1. While the routing table with OSPF is only based on the original topology, this table with MRC is based on the original topology and on all backup topologies. In Fig. 2.1, only router IDs are registered on routing tables for intuitive understanding of relationship between backup topology and routing table. Therefore, the number

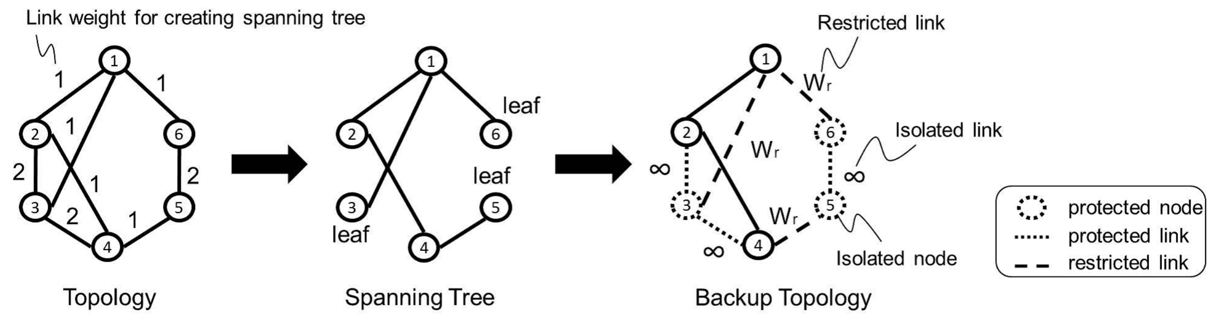


Figure 2.2: Key idea of our algorithm. Spanning tree is created from topology. Backup topology is created from spanning tree. Links removed to create spanning tree become protected links. Links connected to leaf node become restricted links. Leaf nodes become protected nodes.

of entries on one table seems to be proportional to the number of nodes in the network. On the other hand, in an actual network, each router should maintain route prefix for router interfaces and external routes as destination IP address. There are typically many routing prefixes in the range of several thousand [40]. In IP fast reroute techniques, a large table is required with its size proportional to the number of backup topologies, and then required memory size of routers becomes huge. Minimizing the number of backup topologies is therefore required to ensure scalability.

For the applicability to actual network, limitation of packet header size is the main issue. In IP fast reroute techniques, the identifier of the selected backup topology is transmitted by a field, such as ToS field, in IP header. However, the field size is finite, and operator would like to restrict the use of its space for differentiated services codepoint [41]. The available number of bits in an IP header is determined in an actual network. If the required number of backup topologies exceeds the number that can be expressed with the available bits, fast recovery cannot be implemented. Minimizing the number of backup topologies is also required in this sense.

From the above observation, the problem we want to first solve is to minimize the number of backup topologies for ensuring network scalability and applicability to actual condition. Cicic et al. [25] proposed an algorithm to reduce the number of backup topologies. However, their algorithm requires addition of another information table, called a last-hop recovery table, for fast recovery. It is therefore not suitable for the current multiple topology routing architecture [39]. In this chapter, we introduce our previous work [26] that minimize the number of backup topologies only by improving backup topologies design algorithm.

2.3 Scalable Backup Topologies Creation

In this section, we present an overview of our algorithm, and then describe it in detail.

2.3.1 Overview of Our Algorithm

The key idea of our algorithm is that each backup topology is made so that the topology, excluding the protected links, becomes a spanning tree (Fig.2.2). The spanning tree of a given topology is a minimal set of links that connects all nodes. We use a spanning-tree-based approach for the following reason. To reduce the number of backup topologies necessary, each backup topology should protect as many links as possible. Therefore, a backup topology should be a spanning tree, and a spanning-tree-based topology can maximize the number of protected links in one topology.

The next sub-problem is how to generate the minimal set of spanning trees that protects all links. To solve this sub-problem, overlapping of protected links in each backup topology should be minimal. Therefore, we prioritize the protection of non-protected links by link-weight manipulation. We create a minimum spanning tree by the Kruskal algorithm [42]. A minimum spanning tree is a spanning tree whose summation of the link weights is minimal. Therefore, the weights of non-protected links are set to a higher value. The links with higher weights then tend to be removed at the time the spanning tree is created. Therefore, the links with higher weights tend to be protected with priority.

Figure 2.3 shows an overview of our algorithm. The processes in steps 1 and 2 are the key points shown in Fig. 2.2. The process in step 3 checks the end condition. If all nodes and links are protected or number of backup topologies becomes maximum number, a set of backup topologies is produced. The process in step 4 determines the set of weights for creating the next optimal spanning tree. This process needs L iterations. For a performance gain, M iterations can be applied to our algorithm in step 5. In the next section, we describe our algorithm in detail.

2.3.2 Algorithm

Our algorithm automatically creates backup topologies from an undirected weighted graph, G, and the maximum number of backup topologies N. N is used when the available number of backup topologies is restricted. The details of our algorithm are shown in Table 2.2, using the notation shown in Table 2.1. The following process is continued until the set of backup topologies that satisfy the characteristics shown in Section 2.2.1 is created or the number of backup topologies exceeds N.

In step 1, parameters are initialized (lines 1–5), and a set of initial link weights are determined (line 4). Our algorithm randomizes the initial weight value for the iteration occurring in step 5. It should be noted that the set of weights for creating the spanning tree is different from the set of metrics configured in the backup topologies.

In step 2, the spanning tree is created on the basis of a set of given weights, and then one backup topology is created from this spanning tree (lines 7–16). We create a minimum spanning tree with the Kruskal algorithm. A minimum spanning tree is a spanning tree whose summation of link weights is minimal. Therefore, the links with higher weights tend to be removed. The leaf nodes in the spanning tree

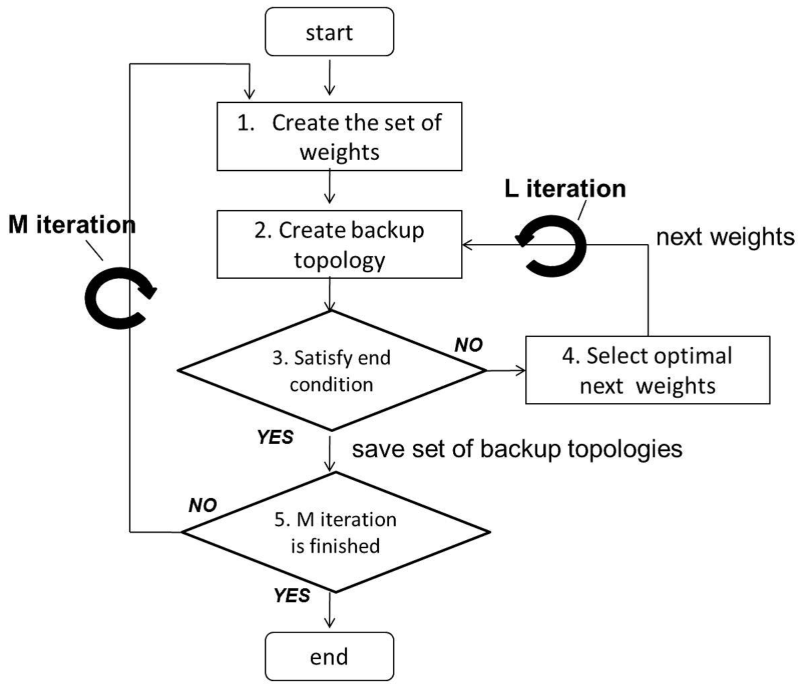

start
1. Create the set of weights
2. Create backup topology
L iteration
next weights
M iteration
3. Satisfy end condition
NO
4. Select optimal next weights
YES
save set of backup topologies
NO
5. M iteration is finished
YES
end

Figure 2.3: Flowchart of scalable backup topology design algorithm

then become protected nodes (lines 8–10). The links that are removed to create the spanning tree become the protected links (lines 11–13). The links that are connected to the leaf nodes in the spanning tree become the restricted links (lines 14–16).

In step 3, the end of our algorithm is evaluated (lines 20–24). Our algorithm maintains a graph, G', which consists of non-protected links and nodes. If G' is empty, our algorithm goes to line 38 (line 21). If a backup topology whose protected nodes and links are all elements in G' can be created, our algorithm creates such a backup topology and goes to line 38 (line 24). If the available number of backup topologies is restricted by N and the current number of backup topologies equals N, our algorithm should stop generating next backup topologies, even if all nodes and links are not protected, and goes to line 38.

In step 4, as mentioned earlier, we solve the sub-problem of discovering the minimal set of spanning trees that protects all links by link weight manipulation (lines 26–36). First, K sets of weights are created. With each new set of weights, we prioritize the isolation of nodes and links that have not been protected. The weights of the non-protected links in the previous backup topology are increased by a random value (line 30). This process protects non-protected links. Next, the weight of one link connected to a non-protected node is set to a lower value, and the weights of other links connected to it are set to a higher value (line 32). This process ensures that the non-protected node becomes a leaf node in the spanning tree. Figure 2.4 shows examples of these processes. The best set, i.e., the set with the most new protected nodes and links, is selected (lines 34–36), and then the process starts again from the creation of a backup topology (step 2). This iteration

Table 2.1: Notation for scalable backup topology design

p	identifier of backup topologies
m	index number for M iteration
n	index number for selecting best set
M	maximum number for M interation
K	maximum number for iteration for selecting best set
G	graph
$V(G)$	set of all vertices in G
$E(G)$	set of all edges in G
$W_{org}(e)$	original metric of link e in graph G
$W(p,e)$	metric of link e in backup topology p
$W^n_{mst}(p,e)$	weight of link e for creating spanning tree in p
W_r	metric of restricted links (high value)
$MST^n(p)$	minimum spanning tree of G based on $W^n_{mst}(p,e)$
RAND()	returns random value from W_{min} to W_{max}
COPY(X)	returns a copy of X
DEL_ ELEMENT(x,X)	deletes element x from set X
NEW_ PROTECTED(T)	number of new protected node (link) by tree T

process from step 4 to step 2 is performed L times. If maximum number of backup topology N does not be given, final value of L becomes maximum number of while loops, that is described in section 2.3.3. If maximum number of backup topology N is given, final value of L becomes $N-1$ because n times iteration produces $n+1$ backup topologies.

In step 5, we iterate our algorithm (lines 38–44). In our algorithm, the form of the first spanning tree is determined by a fixed set of weights. This set of weights affects the generation of subsequent spanning trees that protect the non-protected components. However, sometimes this first set may not be appropriate for generating the subsequent set. Therefore, we iterate our algorithm M times with different initial sets of weights (lines 38-41). Then, the best combination of backup topologies, i.e., the combination with minimal backup topologies, is selected (line 43).

Table 2.2: Algorithm for scalable backup topology design

```
1   $p \Leftarrow 0, optid \Leftarrow 0$ /* optid is ID of optimal weights set */
    $m \Leftarrow 0$ /* index number for M iteration */
2   for all $e \in E(G)$ do
3       $W(p,e) \Leftarrow W_{org}(e)$
4       $W_{mst}^{optid}(p,e) \Leftarrow RAND()$
5   $G'$=COPY($G$)
6   while (1) {
7       T $\Leftarrow MST^{optid}(p)$
8       for all $v \in V(T)$ do
9           if ($v$ is leafnode) then /* search protected node */
10              DEL_ ELEMENT($v$, $V(G')$)
11      for all $e \in \overline{E(G) \cap E(T)}$ do /* search protected link */
12          $W(p,e) \Leftarrow \infty$
13          DEL_ ELEMENT($e$, $E(G')$)
14      for all $e \in E(T)$ do /* search restricted link */
15          if ($e$ is connected with leaf node) then
16              $W(p,e) \Leftarrow W_r$
17      $p++$
18      for all $e \in E(G)$ do /* initialize next metrics */
19          $W(p,e) \Leftarrow W_{org}(e)$
20      if ($G' == \emptyset$) then
21          goto line 38
22      if ($G'$ satisfy backup topology condition) then
23          create $W(p,e)$ based on $G'$
            (all members of G' are protected)
24          goto line 38
25      else /* create K sets of weights, then select best set */
26          for ($n = 0, newoptid = 0; n < K; n++$) do
27              for all $e \in E(G)$ do
28                  $W_{mst}^{n}(p,e) \Leftarrow W_{mst}^{optid}(p-1,e)$
29                  if ($W(p-1,e) \neq \infty$) then
30                      $W_{mst}^{n}(p,e) += RAND()$
31              for all $e$ connected with node $v \in V(G')$ do
32                  weight of one link is set to low value, and
                    weights of other links are set to a high value
33              $T^n \Leftarrow MST^n(p)$
34              if ($n \geq 1$ and (NEW_ PROTECTED($T^n$)
                $\geq$ NEW_ PROTECTED($T^{newoptid}$)))
35                  $newoptid \Leftarrow n$
36          $optid \Leftarrow newoptid$
37  }
38  if ($m < M$) then
39      save current combination of backup topologies
40      $m++$
41      goto line 2
42  else
43      select the best combination of backup topologies
44      end
```

2.3.3 Complexity of Our Algorithm

If the available number of backup topologies is infinity, our algorithm continues until all nodes and links are protected in at least one backup topology. Because of the processes of lines 30 and 32 in Table 2, at each step of a while loop, at

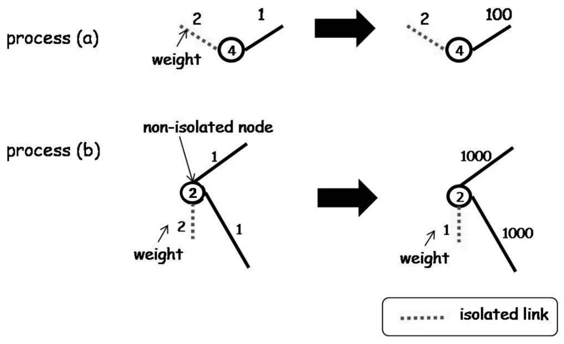


Figure 2.4: Example of our algorithm. (a) Processes in lines 29–30 of Table 2.2. (b) Processes in lines 31–32 of Table 2.2.

least one link or node is protected. Thus, the maximum number of while loops (L iterations) is $|V(G)| + |E(G)|$. The computational complexity of Kruskal algorithm is $|E(G)|log(|V(G)|)$ [42], therefore the worst computational complexity of our algorithm is $(|E(G)| + |V(G)|)K|E(G)|log(|V(G)|)$. Computation simplified under the assumption of $|E(G)| \simeq |V(G)|$, with other parameters aggregated as a constant α, is described as $O(\alpha|V(G)|^2 log(|V(G)|))$. The computation of the conventional algorithm [24] under the same assumption is described as $O(\alpha|V(G)|^2)$. Our algorithm is relatively complex compared to the conventional one. However, it is reasonable to suggest that the computational time of our algorithm, which should be computed offline, is acceptable for practical use.

2.4 Performance Evaluation

In this section, we demonstrate the effectiveness, applicability, and overhead of our algorithm through extensive simulation. First we describe the simulation conditions in section 2.4.1, then we show the minimum number of backup topologies compared to the conventional algorithm [24] for various types of topologies and numbers of nodes in section 2.4.2. Then we discuss the applicability of our algorithm to an actual condition in section 2.4.3. We assume that the available number of backup topologies is restricted for that actual condition. For example, the ToS field size for propagating the ID of a backup topology is finite, and the size of the router memory is also finite. Therefore, we evaluate the ratio of the number of flows that are rapidly recovered under the limited number of backup topologies. Finally, in section 2.4.4, we evaluate the characteristics of backup routes for clarifying the overhead of our algorithm.

2.4.1 Simulation Conditions

In our evaluation, we mainly use two different topology models in terms of node degree variance. The node degree expresses the number of links connected to a node. One model is the Waxman model, whose node degree is relatively uniform on each node. The other model is the Barabasi-Albert (BA) model, whose node degree follows a power-law. In a power-law network, most nodes have a small number of links, while a small portion of nodes have a large number of links. We created 20 instances of topologies for each of these two models using the BRITE topology generation tool [43]. Our evaluation conditions assume point-of-presence (POP) level networks. Therefore, the average node degree d_a is set to two (nodes have four links on average) and is calculated as follows;

$$d_a = |E(G)|/|V(G)| \tag{2.1}$$

$|V(G)|$ is the total number of nodes, and $|E(G)|$ is the total number of edges.

The evaluation index in section 2.4.2 is the minimum number of backup topologies. We also evaluate the effectiveness where available number of backup topologies is limited in section 2.4.3. This situation may occur when other application such as Diffserv also uses ToS field [41]. We define the fast recovery ratio (FR) as an index. FR is calculated as follows;

$$FR = \sum_{e \in E_{protected}(G)} T(e) / \sum_{e \in E(G)} T(e), \tag{2.2}$$

where $T(e)$ is the number of traffic flows on link e, $E(G)$ is the set of all links, and $E_{protected}(G)$ is the set of all protected links in each backup topology. For example, the FR value increases if the number of protected traffic flows increases. If all links are protected, the FR value becomes 1.

Other simulation conditions are as follows. The parameters of our algorithm were set such that there are 20 iterations (M=20) and K=5 (Table 2, line 26). In section 2.4.4, a traffic matrix is required for evaluating the link loads. We use the gravity traffic model [46], whose weight is proportional to the node degree.

2.4.2 Minimum Number of Backup Topologies

Figure 2.5 shows the results for the Waxman model, and Fig. 2.6 shows the results for the BA model. These results suggest that the number of backup topologies obtained with our algorithm will never exceed that obtained with the conventional algorithm [24]. There was a reduction of up to 56% for the BA model with 200 nodes, while there was a reduction of up to 28% for the Waxman model with 200 nodes.

Our algorithm is more effective with large power-law networks (BA model). This is because the BA model has more nodes with a low node degree. To briefly explain, the conventional algorithm [24] selects the target node and maximally protects the links that connect to the target node within the range where the connectivity is kept. If the node degree of a target node is low, the number of links that can be protected decreases. Therefore, if the ratio of nodes with low node degree increases, the number of backup topologies increases because the total number of protected links in one backup topology decreases. By contrast, in our algorithm, dependency on the distribution of the node degree is low because it creates backup topologies one at a time by using the spanning tree. Therefore, a

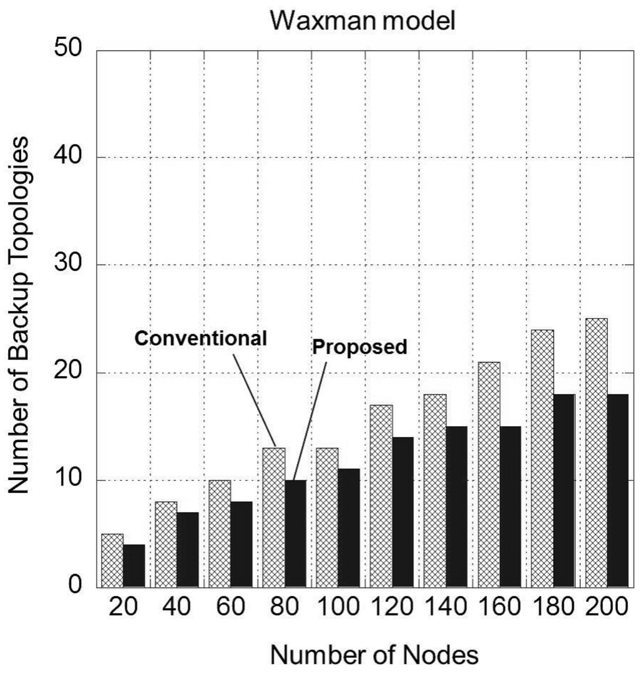


Figure 2.5: Number of backup topologies in Waxman model. Average node degree is 2.

fixed number of links is always protected. If the number of nodes increases, the ratio of nodes with low node degree also increases. Therefore, the effectiveness of our algorithm increases by the same reason.

From these results, we believe our algorithm will be effective on an actual network. This is because the maximum number of nodes in an intra-domain is assumed to be about 200 [44]. Moreover, Medina et al. [43] reported that network topologies of major ISPs often show a power-law tendency. Therefore, our evaluation shows the effectiveness on a large actual network though the results for real-life network could not be reproduced here for proprietary reasons.

2.4.3 Applicability to Actual Condition

Figure 2.7 shows the fast recovery ratio for the BA model under a limited available number of backup topologies. In all conditions, our algorithm requires only 3 backup topologies to achieve FR = 1.0, while the conventional algorithm [24] requires from 6 (20 nodes) to 12 (80 nodes) backup topologies. In addition, even if the available number of backup topologies is limited to up to two, the recovery ratio with our algorithm remains at more than 0.9, while with the conventional algorithm, it decreases to 0.17 in the worst case (80 nodes). The results for the Waxman model have almost the same tendency [26].

The results also suggest that there is low dependency between the required number of backup topologies for achieving FR = 1.0 and the number of nodes. This is because the ratio of protected links in one backup topology does not depend on the number of nodes. The set of links $E_r(G)$, which is removed from the connected

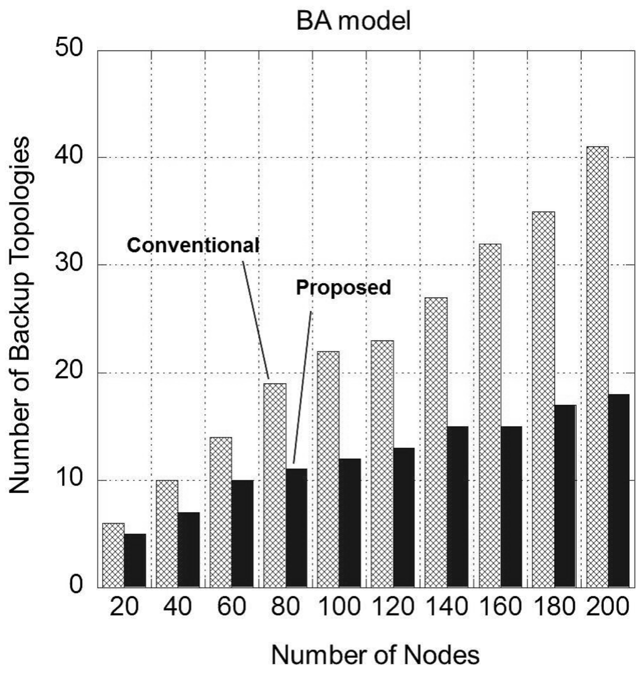


Figure 2.6: Number of backup topologies in Barabasi-Albert (BA) model. Average node degree is 2.

graph to create the spanning tree, is described as follows [45]:

$$|E_r(G)| = |E(G)| - |V(G)| + 1. \tag{2.3}$$

Therefore, the ratio of non-protected links to all links in one backup topology is described as follows;

$$\frac{|E(G)| - |E_r(G)|}{|E(G)|} \quad = \quad \frac{1}{d_a}\left(1 - \frac{1}{|V(G)|}\right). \tag{2.4}$$

By (2.4), we found that the value of d_a is dominant. If $V(G)$ becomes infinity, (2.4) converges on the reciprocal of d_a. Therefore, the number of protected links mainly depends on d_a.

2.4.4 Characteristics of Backup Routes

In this section, we discuss the characteristics of backup routes in terms of number of hops and link loads. Our algorithm aggregates protected links in one backup topology to reduce the number of backup topologies. Therefore, the number of links that are used as backup routes decreases. This may cause high link loads on certain links or an increase of the hop count of backup routes. Therefore, we quantitatively clarify the characteristics of backup routes. For the BA model whose reduction effect on the number of backup topologies is high, the number of hops and the link load adversely increase more than for the Waxman model. Therefore, we evaluate on the BA model in order to clarify the disadvantage. Although we

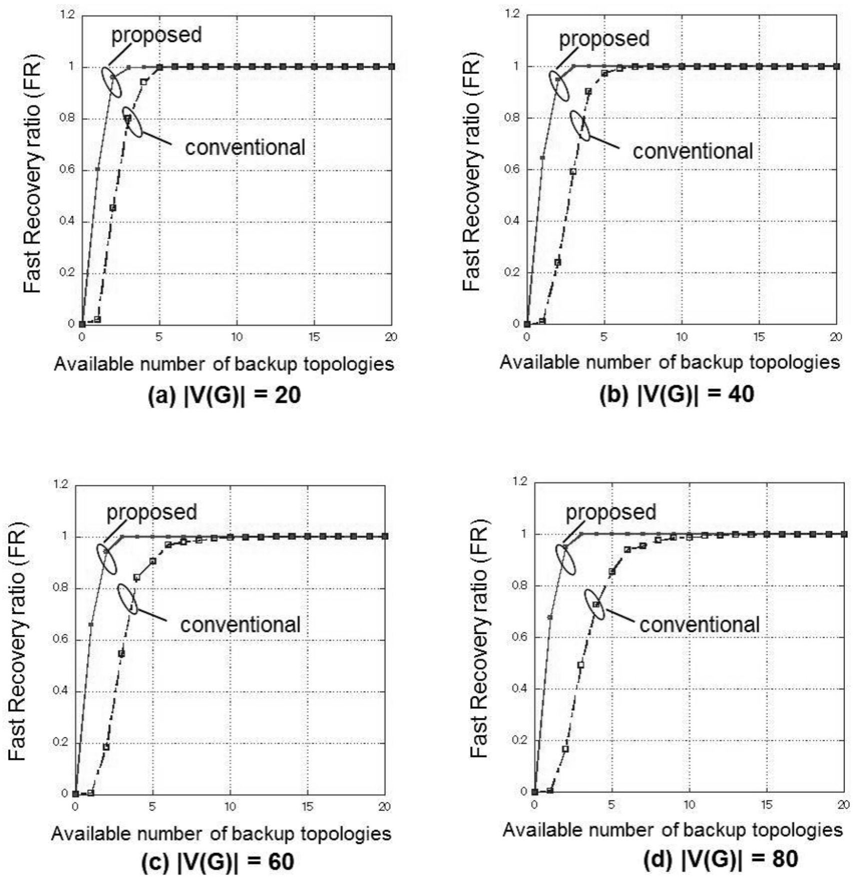


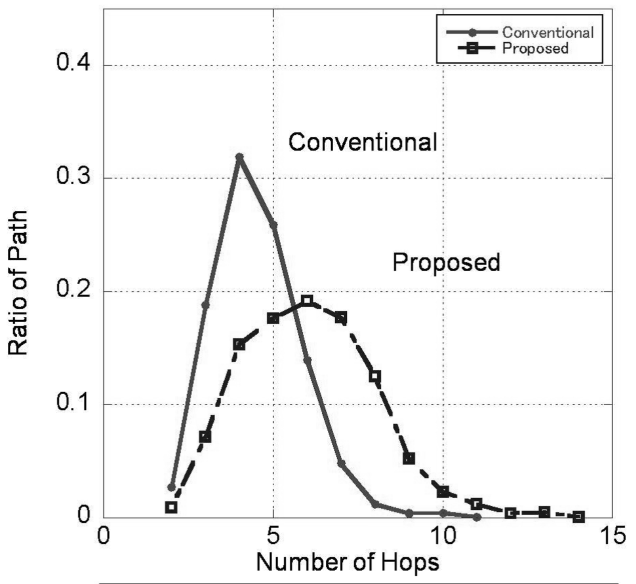
Figure 2.7: Fast recovery ratio in BA model. Average node degree is 2. $|V(G)|$ is number of nodes.

only show the results of a 40-node topology, the results for other numbers of nodes had almost the same tendency.

Figure 2.8 shows the distribution of the hop counts of flows that are affected by the failure, and Fig. 2.9 shows the distribution of the hop counts of all flows. When we focus on the flows that are affected by the failure, the hop counts averagely increase, and the maximum hop count of the backup routes increases by three hops. In contrast, the distribution of the hops of all flows is almost the same as in the conventional algorithm [24].

Figure 2.10 shows the link loads of all links in the worst case. From this result, we can calculate the required link bandwidth for avoiding congestion by adding all the link loads. When the required link bandwidth of the conventional algorithm is standardized by 1.00, that of our algorithm becomes 1.04. That is, the total bandwidth remains almost the same, but with a slight increase.

Although the number of hops or link loads of our algorithm increases compared to that of the conventional one, our results show that these impacts are relatively low. This is because the backup routes, which are calculated with the backup topologies, are used only for traffic crossing a failed resource. The traffic that does not use the failed resource is forwarded in accordance with the original topology. In our evaluation, only about 3.6 % of traffic flows was affected by the failures.



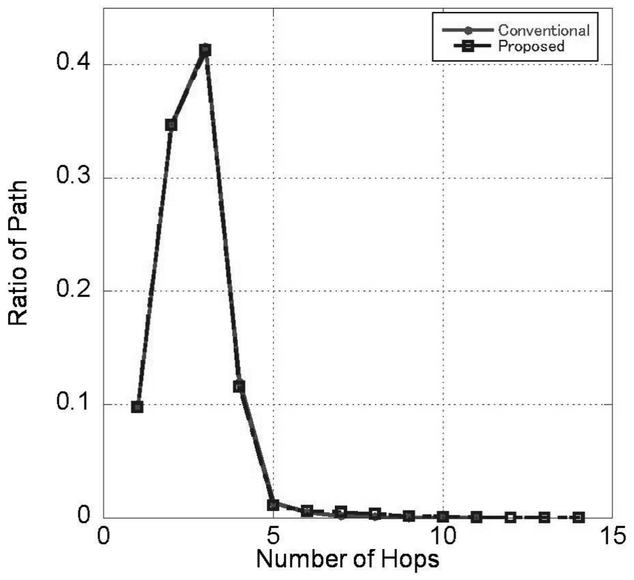
	Conventional	Proposed
Minimum Hop	1	1
Maximum Hop	11	14
Average Hop	4.5	6.0

Figure 2.8: Number of hops of flows that are affected by the failure (40-node BA model).

Other traffic flows were not affected. Moreover, these backup routes are only used until the new backup routes are calculated by the link-state protocol. Therefore, we believe these adverse characteristics of backup routes are not as important as the number of backup topologies required.

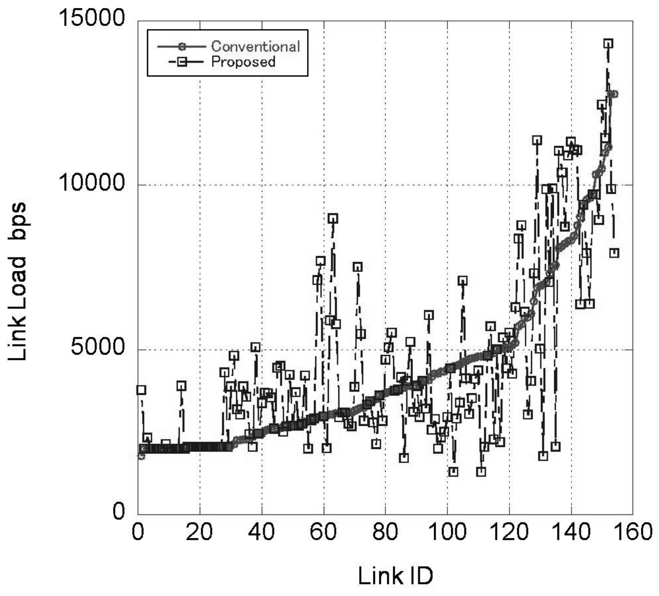
2.5 Conclusion

We argued that minimizing the number of backup topologies for IP fast rerouting is required to improve network scalability and applicability. We presented a new backup topology computation algorithm that reduces the number of backup topologies. Our algorithm has high feasibility for the existing IP fast rerouting framework in terms of the memory consumption, and we demonstrated its effectiveness through an extensive simulation study. The results showed that our algorithm reduced the number of backup topologies by about 56% on power-law networks, and the effectiveness increases as the number of nodes increases. Our algorithm is more effective on actual networks than the conventional algorithm.



	Conventional	Proposed
Minimum Hop	1	1
Maximum Hop	11	14
Average Hop	2.63	2.67

Figure 2.9: Number of hops of all flows. (40-node BA model).



	Conventional	Proposed
Required Link Bandwidth (bps)	1	1.04

Figure 2.10: Worst-case link loads of all links (40-node BA model). Link ID is sorted by link loads of conventional algorithm.

Chapter 3

Loop-free IP Fast Rerouting considering Double Link Failures

3.1 Introduction

In this chapter, we tackle the IP fast restoration problem from concurrent double failures. The main issue in IP fast rerouting considering multiple failures is avoiding forwarding loops. In the original MRC method and related works [24–26,28,29] considering single failure, forwarding loops occur essentially because the set of backup topologies is created only to protect a single node or a single link failure. Hansen et al. proposed a forwarding algorithm and backup topology design algorithm for double failures [27]. However, its forwarding algorithm allows finite loops to occur. Even if a loop period is finite, it consumes the network resources, and packets are dropped if its time to live (TTL) is expired.

From the above observation, we propose novel packet forwarding algorithm and backup topologies design algorithm that avoids the forwarding loops including finite loops as much as possible in the event of concurrent double failures. For avoiding the forwarding loops, the second failure detecting node should know the first failure point. However, explicit notification of the failure point decreases the scalability of IP fast rerouting. Our key idea is estimating the first failure point from the packet header information. We also present a backup topology design algorithm with less number of backup topologies. We focus on the fact that backup routes can be provided by combining routes from two backup topologies: design restriction, which requires protecting arbitrary two-link pairs on one of the backup topologies, is relaxed. Then, by adding the backup topologies, whose relay links (non-protected links) has a high betweenness centrality, we can maximize the diversity of backup routes.

Our packet forwarding algorithm can reduce the average loop probability from 10^{-2} to 10^{-3} compared to the existing algorithm [27]. Though the optimal solution, which can notify the explicit failure point, can achieve much more reduction of loop probability, its applicability is up to about 60 nodes because of the bit field limitation. For the backup topologies design, our algorithm can reduce the number of backup topologies about 35-50% compared to the loose lower bound solution.

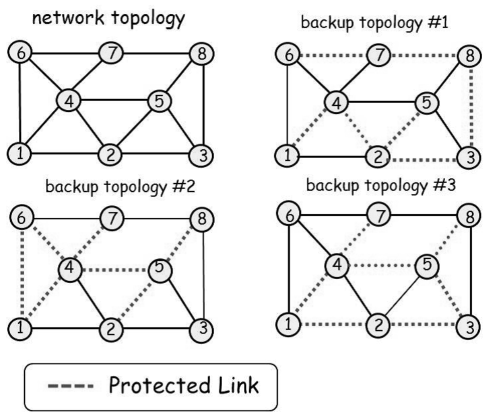


Figure 3.1: Overview of backup topologies. Each backup topologies are created by spanning tree-based algorithm

In addition, a required number of backup topologies for achieving target value of the loop probability decreases as the number of nodes increases: our algorithm has high scalability.

The rest of this chapter is organized as follows. In Section 3.2, we introduce an overview of the IP fast rerouting, related works and our problem statement. Section 3.3 presents our packet forwarding algorithm that does not require explicit failure notification. In Section 3.4, our backup topology design algorithm is presented. Our evaluation results are shown in Section 3.5, and then we conclude our discussion in Section 3.6.

3.2 Forwarding Loop Problem and Problem Statement

In this section, we describe the forwarding loop problem when double failures occur. Then, our problem statement and formulation of the loop-free conditions against double failures is presented.

3.2.1 Forwarding Loop Problem on Double-Link Failures

In this section, we focus on forwarding loop problem when double-link failures occur. If we deploy the existing IP fast rerouting algorithms [24–26, 28, 29] for recovering from double-link failure, forwarding loops stochastically occur. Figure 3.2 illustrates examples based on backup topologies in Fig. 3.1. We assume that link x and link y simultaneously fail when packets are forwarded from node *src* to node *dst* in each case ((i) - (iii)). In the case (i), node i selects backup topology #1 because it protects link x, and then tries to forward the packets to nodes 7-4-6. Then, node j detects failure y, and selects backup topology #2, which protects link y. The route destined for node *dst* from node j is 4-2-3-8-7-6, and it includes link x. A node i also selects backup topology #1 and then forwards packets to node j. A forwarding loop including nodes i and j therefore results.

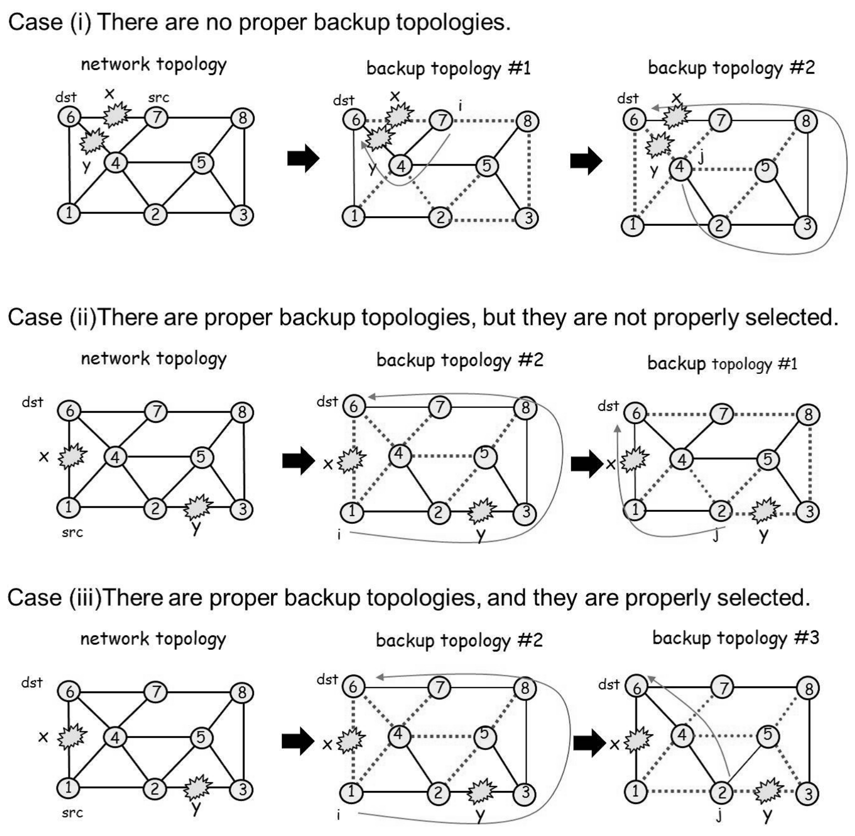
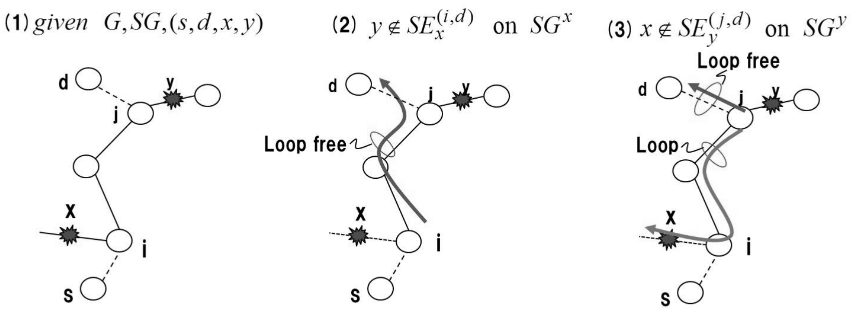

Case (i) There are no proper backup topologies.
network topology
backup topology #1
backup topology #2
dst
src
x
y
i
j
Case (ii)There are proper backup topologies, but they are not properly selected.
network topology
backup topology #2
backup topology #1
dst
src
x
y
i
j
Case (iii)There are proper backup topologies, and they are properly selected.
network topology
backup topology #2
backup topology #3
dst
src
x
y
i
j

Figure 3.2: Examples of forwarding loops when double failure occurs. In each cases (i)-(iii), the node that detects failure x is noted as i, and the node that detects failure y is noted as j. The link metric of each link is assumed to be 1 for shortest path computation.

In this case, there is no proper backup topology to detour the failures without causing a forwarding loop. The case (ii) shows another loop scenario whose failure points, source and destination are different from case (i). A node i selects backup topology #2, and node j selects backup topology #1 for avoiding failure links x and y respectively. In this case, a forwarding loop including nodes i and j also occurs. On the other hand, if node j selects backup topology #3, which also protects failure y, the forwarding loop is avoided (case (iii)). That is, for avoiding forwarding loops, both a proper backup topology design and a backup topology selection (forwarding) algorithm are required.

Hansen et al. [27] proposed an IP fast rerouting algorithm, which recovers from double-link failures. It constructs backup topologies where arbitrary two links are protected on one of the backup topologies. If a backup topology is properly selected, arbitrary double-link failure can be avoided by the selected backup topology. On the other hand, failure detecting node does not know the first failure point because only the forwarding table ID is notified to each node. Their algorithm therefore repeats backup topology selection until proper backup topology is found: the algorithm allows finite loops to occur. Even if a loop period is finite, it consumes the network resources, and packets are dropped if their time to



(1) given $G,SG,(s,d,x,y)$
(2) $y \notin SE_x^{(i,d)}$ on SG^x
(3) $x \notin SE_y^{(j,d)}$ on SG^y
d
y
j
x
i
s
Loop free
Loop free
Loop

Figure 3.3: Loop-free condition providing protection from double-link failure.

live (TTL) is expired. In addition, protecting against a double-link failure by one backup topology is a redundant approach in terms of the number of backup topologies. As shown in Fig. 3.2 (iii), a loop-free route is provided by the combination of backup topologies.

3.2.2 Problem Statement

From the above observations, our goal is to minimize the loop occurrences as much as possible in the case of double-link failures. As shown in Fig. 3.2 (ii), inadequate packet forwarding causes a loop even if a set of backup topologies has backup route against double failures. Then, our challenge compared to the existing works [27] is establishing a novel forwarding algorithm, which deterministically avoids forwarding loops including finite loops. As another loop scenario, there is possibility that any combination of backup topologies cannot provide backup route which avoids forwarding loop (Fig. 3.2 (i)). Then, we also define the backup topologies design problem with less number of backup topologies.

For the deep analysis, we formulated the loop-free condition that provides protection against double-link failures. A network is given as a directed graph $G = (V, E)$. Here, V is a set of nodes, and E is a set of links. The set of backup topologies defined in [24] is denoted as $SG = (SG_1, SG_2, ..., SG_k)$, $SG_k = (V, E_k)$. We also define the set of non-protected links as $E_k^{np} \in E_k$. That is, E_k^{np} is composed of the links for packet forwarding. The source node is denoted as $s \in V$, and the destination node is denoted as $d \in V$. The first failure is $x \in E$, and the second failure is $y \in E \setminus x$. Note that the first failure x occurs at either of the links that composes the original shortest path. If we define time $T(x)$ as the time when packets encounter failure x, $T(x) < T(y)$ is always satisfied. Nodes $i \in V$ and $j \in V$ respectively detect failures x and y. The backup topology that protects link x is denoted as $SG^x = (V, E^x) = \{SG_k \in SG : x \notin E_k^{np}\}$, and the subset of E^x that composes the route from node a to node b is denoted as $SE^x_{(a,b)} \subset E^x$. Then, if the following condition is satisfied for the arbitrary (s, d, x, y), the given G and SG can provide the loop-free backup routes that provide protection from arbitrary double-link failures (Fig. 3.3).

$$y \notin SE^x_{(i,d)} \vee x \notin SE^y_{(j,d)} \tag{3.1}$$

Table 3.1: Notation of Algorithm1

$G = (V, E)$	: Graph with nodes V and undirected links E
$SG = (SG_1, ..., SG_k)$	: A set of backup topologies
$nextFib(s, d, y, f_{in})$	: Pointer to f_{out}, which is for given (s, d, y, f_{in}) tuple
$sPathLinks(n, d, G)$	: Return a set of links, which composes shortest path from s to d in G
$pLinks(SG_i)$	: Return a set of protected links in SG_i

.

Table 3.2: Algorithm1: packet forwarding without failure notification. Above figure illustrates the BTS table creation for node n.

```
1  For all $(s, d, y)$ tuple in node $n$
2      $nextFib(s, d, y, 0) := i$, $y$ is protected in $SG_i$ &&
       (|sPathLinks($n, d, SG_i$)| == minimum)
3      E1 := $sPathLinks(s, d, G)$
4      For (k=1; k $\leq$ |SG|; k++)
5          E2 := pLinks($SG_k$)
6          E3 := E1 $\cap$ E2
7          If (E3 == $\emptyset$)
8              continue
9          $nextFib(s, d, y, k) := i$, $y$ is protected in $SG_i$ &&
           |sPathLinks($n, d, SG_i$) $\cap$ E3| == minimum
```

.

3.3 Packet Forwarding Algorithm without Failure Notification

This section presents a packet forwarding algorithm, which avoids forwarding loop without explicit failure notification. To realize the loop-free backup route against double failures, the second failure-detecting node should be able to detect the first failure. However, with the standard protocol using backup topologies [39], there is a restriction in the failure notification: the failure-detecting node only provides information on the backup topology ID. That is, it cannot directly provide information on the failure link or node.

We propose a packet forwarding algorithm, which estimates the first failure point from the packet header information, and avoid it as much as possible. We prepare the backup topology selection (BTS) table in each node for selecting the proper backup topology when a node detects a failure. The failure detecting node refers to its BTS table and selects one of backup topologies for rerouting. Then, by properly designing the BTS table, we can control the loop occurrences.

The details of our BTS table creation algorithm are shown in Table 3.2, using

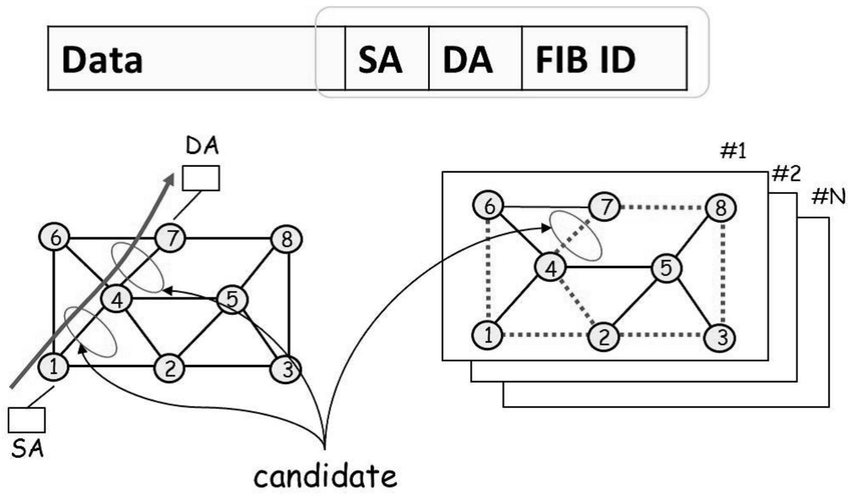


Figure 3.4: Example of failure point estimation.

the notation shown in Table 3.1. This algorithm outputs the backup topology ID f_{out} for rerouting when second failure y, source IP address s, destination IP address d, and input backup topology ID f_{in}, which is designated in ToS field, are inputted on a node n. The backup topology ID is defined as a sequence number which is from 0 to 255. The ID of packets, which do not encounter the failure, is marked as 0.

Algorithm iteratively performs the following processes for each (s, d, y) tuple (line 1). Firstly, it determines f_{out} for the packet which do not encounter the failure ($f_{in} = 0$). The backup topology i, whose route can avoid failed link y and provides shortest path from n to d is minimal, is selected, and $nextFib(s, d, y, 0)$ is set as i (line 2). The shortest path provides the routes, whose probability to encounter the other failure is minimal. Then, we compute the original shortest path from s to d, and extract the links that constitute the original shortest path as $E1$ (line 3). Our algorithm, then, assumes the possible incoming of f_{in}, and performs the following processes (line 4). It extracts the links that are protected on the backup topology SG_k as $E2$(line 5). Through logical multiplication of $E1$ and $E2$, we get the assumed failed links as $E3$ (line 6). If $E3$ is empty set, this entry creation is skipped because such a routing scenario does not exist (line 7, 8). Otherwise, the backup topology i, whose route can avoid failed link y and have less inclusion of $E3$, is selected, and $nextFib(s, d, y, k)$ is set as i (line 9). When $|E3|$ equals to one, failure point is uniquely identified. Thus, forwarding loop is avoided if proper backup topology exists. For the case of Fig. 3.4, we can identify the failed link as link 4-7, and then we only have to select the backup topology that does not use the second failed port and the first failed link 4-7.

Noted that the BTS table is created offline, so computation of the above algorithm does not affect the real-time rerouting [47]. The size of BTS table is proportional to the number of (s, d) pairs. Thus, this paper does not discuss the size of BTS table because compression of BTS tables depends on the IP addresses aggregation.

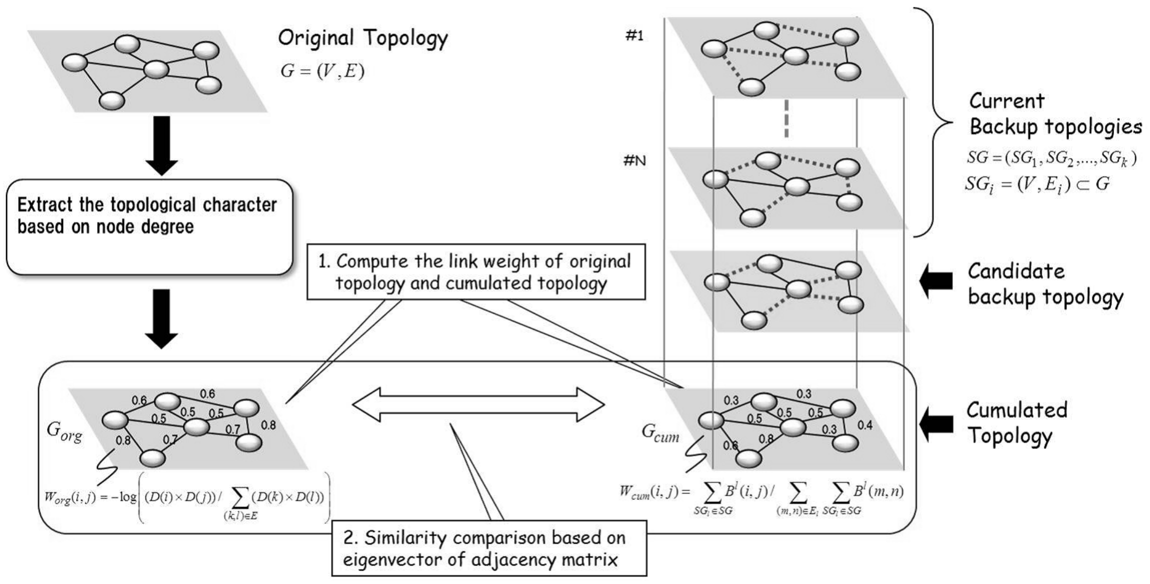

Original Topology
$G = (V, E)$
Extract the topological character based on node degree
1. Compute the link weight of original topology and cumulated topology
#1
#N
Current Backup topologies
$SG = (SG_1, SG_2, \ldots, SG_k)$
$SG_i = (V, E_i) \subset G$
Candidate backup topology
Cumulated Topology
G_{org}
G_{cum}
$W_{org}(i,j) = -\log\left((D(i) \times D(j)) / \sum_{(k,l) \in E} (D(k) \times D(l)) \right)$
$W_{cum}(i,j) = \sum_{SG_i \in SG} B^l(i,j) / \sum_{(m,n) \in E_i} \sum_{SG_i \in SG} B^l(m,n)$
2. Similarity comparison based on eigenvector of adjacency matrix

Figure 3.5: Similarity comparison-based backup topology design algorithm

Existing forwarding mechanism [27] uses backup topology ID-order random selection. In addition to the existing forwarding mechanism, the following algorithms are used to compare the performance of our algorithm. The performance comparison is presented in Section 3.5.

1. **Shortest & Random:** To reduce the probability of encountering the next failure, it selects the backup topology that protects the failed port and provides the shortest path at the first failure detection. And a backup topology is randomly selected at the second failure detection.

2. **Shortest & Shortest:** Its selection scheme is the same as First Shortest at the first failure detection, and it also selects the backup topology that provides the shortest path at the second failure detection.

3.4 Backup Topology Design for Protection against Double Failures

In this section, we propose a backup topology design algorithm to reduce the loop probability against the arbitrary double link failure. As described in Sec. II.B, there is possibility that any combination of backup topologies cannot provide backup route which avoids forwarding loop. Our approach is adding the backup topologies to the minimal set of them [26].

A sufficient condition for achieving the loop-free condition is that arbitrary two links are protected on one of the backup topologies, and Ref. [27] adopted such an approach. However, this condition is not necessary and causes redundant backup topologies because the backup route may be constructed by a combination of backup topologies as shown in Fig. 3.2(iii).

From this observation, we focus on increasing the diversity of backup routes that consist of a combination of various backup topologies. To increase the diversity, we add the backup topologies, whose relay links (non-protected links) has a high betweenness centrality. The link betweenness centrality is a measure of a link centrality in a network, and it equals to the number of shortest paths from all vertices to all others that pass through that link. That is, by adding the backup topology, whose relay links (non-protected links) has a high betweenness centrality, the diversity of backup routes with backup topologies increases.

Hereafter, we propose the similarity comparison-based algorithm. Firstly, we extract the betweenness centrality as a characteristic of the original topology. Then, the number of relay links of backup topologies is extracted as a characteristic of the set of backup topologies. Then, by maximizing the similarity between them, we could get the backup topologies, whose relay links has a high betweenness centrality (Fig. 3.5).

3.4.1 Similarity Computation Based on Eigenvector of Adjacency Matrix

We describe the computation of similarity $S(G, SG)$ when the original topology $G = (V, E)$ and backup topologies $SG = (SG_1, SG_2, ..., SG_k)$, $SG_k = (V, E_k)$ are given. First we compute the original topology metrics based on the betweenness centrality of original topology. We denote the link between node i and j as $(i, j) \in E$, and the betweenness centrality of link (i, j) as $B(i, j)$. Then, link metric $W_{org}(i, j)$ of link $(i, j)$is denoted as follows;

$$W_{org}(i,j) = \left(B(i,j)/\sum_{(k,l)\in E} B(k,l)\right)/\left(\sum_{(m,n)} \{B(m,n)/\sum_{(k,l)\in E} B(k,l)\}\right) \tag{3.2}$$

In (3.2), The links, whose betweenness centrality is high, has high value, and each link metric is normalized by sum of link metrics. Then we generate the adjacency matrix G_{org}, and it has above value (3.2) on existing links, and values of other elements are zero. After that, we calculate the principal eigenvector V_{org} of G_{org}, which has the largest eigenvalue. This strategy that regards the principal eigenvector (spectrum) as character of network topology is mature method [48], and the element of this vector is known as an eigenvector centrality that indicates the relative importance of the node in the graph.

Then, we compute the eigenvector based on the set of backup topologies SG. We assign the following binary value to the link $(i, j) \in E_l$ that exists on backup topology SG_l.

$$RL^l(i,j) = \begin{cases} 1 & \text{if link (i,j) is relay link} \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

Then, link metric $W_{cum}(i, j)$ of cumulated topology derived by superimposing backup topologies is computed as follows;

$$W_{cum}(i,j) = \sum_{SG_l \in SG} RL^l(i,j) \tag{3.4}$$

Table 3.3: Algorithm2:Topology adding algorithm. Input information is original topology and current set of backup topologies.

```
1 compute $S(G, SG) \Rightarrow S_{opt}$
2 for $(k = 1; k \leq k_{th}; k++)$
3     generate $SG_k$
4     $SG' := SG \cup SG_k$
5     compute $S(G, SG') \Rightarrow S_k$
6     if $(S_{opt} < S_k)$
7         $S_{opt} := S_k$
8         index $:= k$
9 return $SG_{index}$
```

After that, we generate the adjacency matrix G_{cum}, and it has above value (3.4) on existing links, and values of other elements are zero, and compute principal eigenvector V_{cum} from G_{cum}.

Finally we compute the similarity $S(G, SG)$ from the two eigenvectors V_{org} and V_{cum}. We compute the cosine similarity as follows;

$$S(G, SG) = \frac{V_{org} \cdot V_{cum}}{|V_{org}||V_{cum}|} \tag{3.5}$$

For example, if the link whose betweenness centrality is high on original topology frequently becomes the relay link on backup topologies, similarity increases. This is match with our design.

3.4.2 Topology Design Algorithm

Table 3.3 shows our proposed pseudo code of the backup topology adding algorithm. The input information is an original topology G and a set of backup topologies SG. First the similarity S_{opt} is computed from G and SG. Then a candidate backup topology that increases the similarity is discovered in k_{th} iterations. In line 3, candidate topology SG_k based on a random link cost is generated, and a new set of backup topologies SG', which is composed of SG and SG_k, is created (line 4). Then the similarity S_k is computed from G and SG' (line 5). If S_k is larger than S_{opt}, S_{opt} is updated by S_k, and the index of the candidate that has the optimal value is saved (lines 7,8). After finishing the k_{th} iteration, the candidate topology that provides the largest similarity is output as the new backup topology (line 9).

Note that Table 3.3 only illustrates the basic design of our algorithm, and an approach with stronger optimization such as the metaheuristics approach [49] can easily be applied. For example, the SG_k generation in line 3 may be based on previous information SG_{k-1}: SG_k inherits the good points of SG_{k-1}. In addition, diversification can also be applied in line 3 to avoid the local optimal situation.

3.4.3 Qualitative Characteristics of Algorithm

The computation of similarity is lightweight. Counting out the link metric is dominant, and the computation is $O(|E|)$. If we directly deploy the loop probability as an objective function, we should compute routes that provide protection against arbitrary double failure $O(|E|^2)$ for every node pair $O(|V|^2)$. Therefore, for the topology adding algorithm, $O(|V|^2|E|^2)$ computation is required for every k iteration, and this is not practical.

3.5 Performance Evaluation

Our evaluation objectives are showing the effectiveness of (1) our packet forwarding algorithm and (2) our backup topology design algorithm. Evaluation indexes are (1) loop probability and (2) required number of backup topologies for achieving the objective loop probability. Hence, we introduce the calculation method of the average loop probability. First, we fix the source and destination nodes pair $(s, d)$and first failure link x. The first failure link x is selected from the links that compose the original shortest path for (s, d). Then we generate the second failure y at other existing links, and obtain the loop probability against (s, d, x)tuple. By evaluating the possible (s, d, x) tuple, we can compute the average loop probability.

For the first evaluation (1), we evaluate our algorithm compared to the existing algorithm [27], benchmarks (Shortest & Random and Shortest & Shortest) described in Sec. 3.3, and optimal forwarding. With the optimal forwarding, we assume that first failure point is explicitly notified. For the second evaluation (2), we generate enough number of multiple backup topologies by iteratively performing our design algorithm. Then the minimum needed to achieve the objective loop probability with optimal forwarding and our estimation-based forwarding is selected respectively. A loose lower bound solution is also compared to our algorithm. The loose lower bound solution satisfies the following condition: arbitrary two-link pairs are protected on one of the backup topologies.

Network topology is a Barabasi-Albert model [43], whose node degree follows power-law, and average node degree is set to two. The number of nodes changes from 20 nodes to 100 nodes. The set of initial backup topologies are computed as a minimal set with our existing work [26] described in chapter 2. We also assume that the double failures do not segment the network.

3.5.1 Effectiveness of Packet Forwarding Algorithm

Figure 3.6 illustrates the loop probability with different packet forwarding algorithms on a minimal set of backup topologies. The average loop probability decreases as the number of nodes increases, and our algorithm can reduce the average loop ratio from 10^{-2} to 10^{-3} on 100 nodes network compared to the existing algorithm [27]. The reason the loop probability decreases is that the number of available backup topologies increases as the number of nodes increases. Then, opportunity of proper backup topologies selection also increases. Other interesting

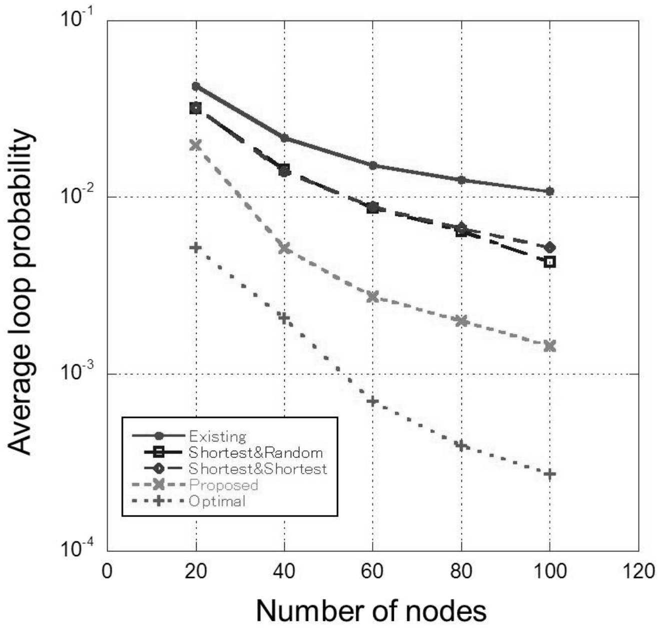


Figure 3.6: Loop probability with minimal number of backup topologies. Numbers of backup topologies from 20 - 100 nodes resulted from spanning tree-based algorithm are 5, 7, 9, 11, 12 respectively.

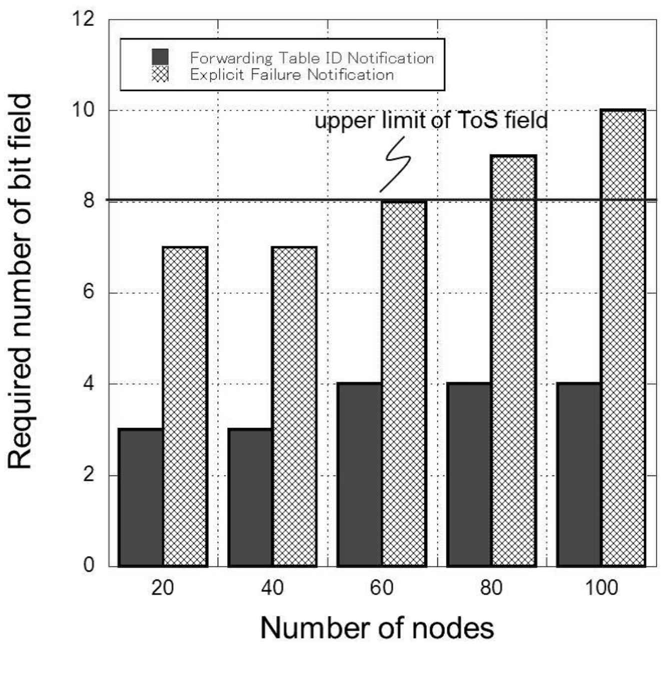


Figure 3.7: Required number of bit field for information notification. Existing algorithm, benchmarks, and our proposal uses forwarding table ID notification. Only optimal forwarding uses explicit failure notification.

result is that Shortest & Random algorithm outperforms Shortest & Shortest algo-

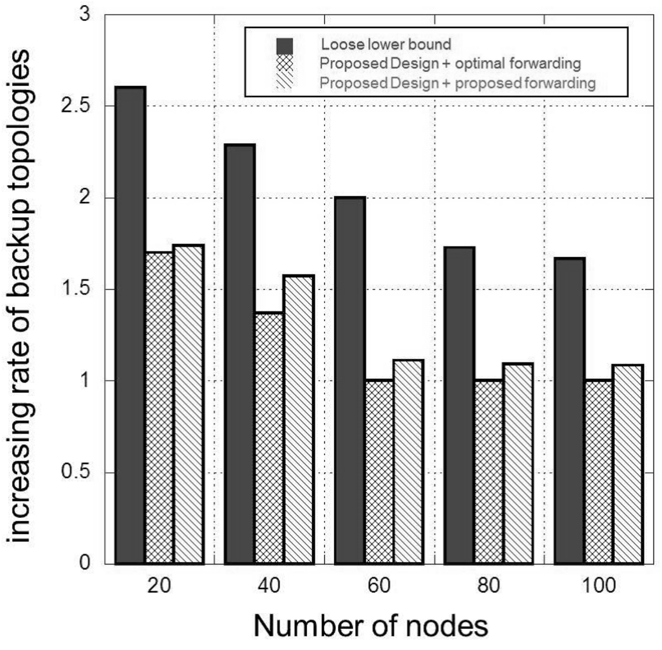


Figure 3.8: Required number of backup topologies. Objective value of the loop probability is set to 10^{-3}.

rithm. If we select the shortest path at the second failure detection, its path tends to include the links on the original shortest path. As a result, it also includes first failure, which is also on the original shortest path.

Though the optimal solution, which can notify the explicit failure point, can achieve further reduction of loop probability, it does not have scalability. Figure 3.7 illustrates the required number of bit field for notifying information. Existing algorithm, benchmarks, and our proposal only notify the ID of backup topologies. Because the maximum number of backup topologies is 12 on 100 nodes network, maximum required number of bit is four. On the other hand, optimal solution should designate the explicit failed link. Then, the amount of information is proportional to the number of links. If we use the ToS field for information notification, applicability of optimal forwarding is limited up to about 60 nodes with our simulation conditions.

3.5.2 Effectiveness of Backup Topologies Design Algorithm

We show the effectiveness of our backup topologies design algorithm in Fig. 3.8. By comparing the loose lower bound solution (left bar) and our backup topologies design (center bar) under the optimal forwarding conditions, we can directly observe the effectiveness of our backup topology design algorithm: it can reduce the number of backup topologies about 35-50% compared to the loose lower bound solution. Noted that objective value of the loop probability is set to 10^{-3}, there is no addition of backup topologies from 60 nodes to 100nodes network. It also suggests that two-link protections is not necessary condition.

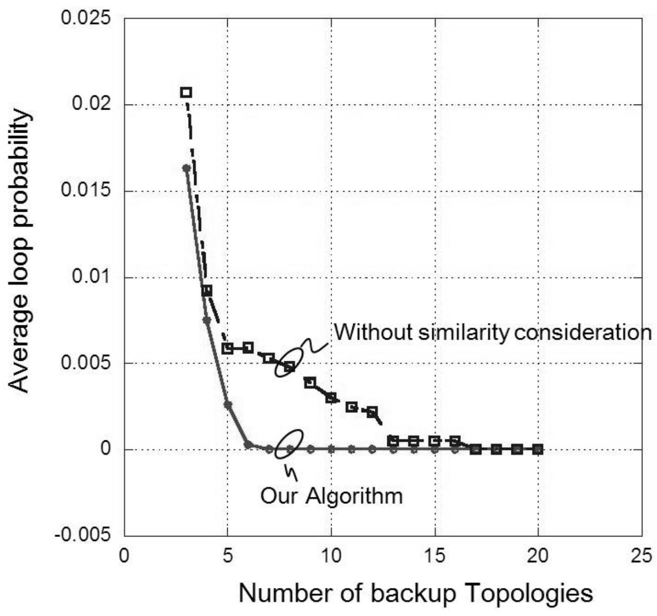


Figure 3.9: Loop probability of cost239 model (11 nodes 25 links).

Though the required number of backup topologies are slightly increases by combining our packet forwarding algorithm (right bar), its impact is relatively low. Especially, a required number of backup topologies for achieving objecitve value of the loop ratio decreases as the number of nodes increases: our algorithm has high scalability. We also evaluate the required number of backup topologies with the existing algorithm and benchmarks. They are probabilistic approach, and then reduction effect of the loop probability is low even if we prepare the enough number of the backup topologies (e.g., over the 50 number of backup topologies): the loop probability with existing algorithm, Shortest & Random, and Shortest & Shortest becomes 2.0%, 0.6%, and 1.2% respectively .

3.5.3 Validity of Backup Topology Design Based on Similarity Comparison

In this subsection, we evaluated the validity of our algorithm based on similarity compared to a simple benchmark without similarity consideration algorithm. This algorithm does not consider the similarity with the original topology. It computes a new backup topology based on the number of relay links in the existing backup topologies. That is, it fairly adds the relay link to each link independent of the node degree of the original topology. Figures 3.9 and 3.10 respectively illustrate the results of the average loop probability and similarity for the cost239 model, and Figs. 3.11 and 3.12 illustrate the results for the 40 nodes power-law model. Noted that the reason the absolute value of the loop probability of power-law model was less than that of cost239 is that the power-law model is larger than the cost239 model: the ratio of failures that do not affect the forwarding loop also increase with the size of networks. From these results, we get the interesting findings about

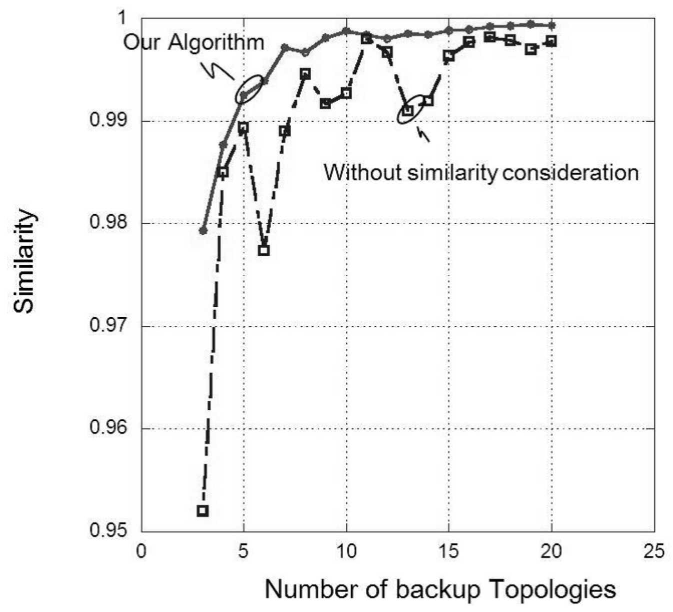


Figure 3.10: Similarity of cost239 model (11 nodes 25 links).

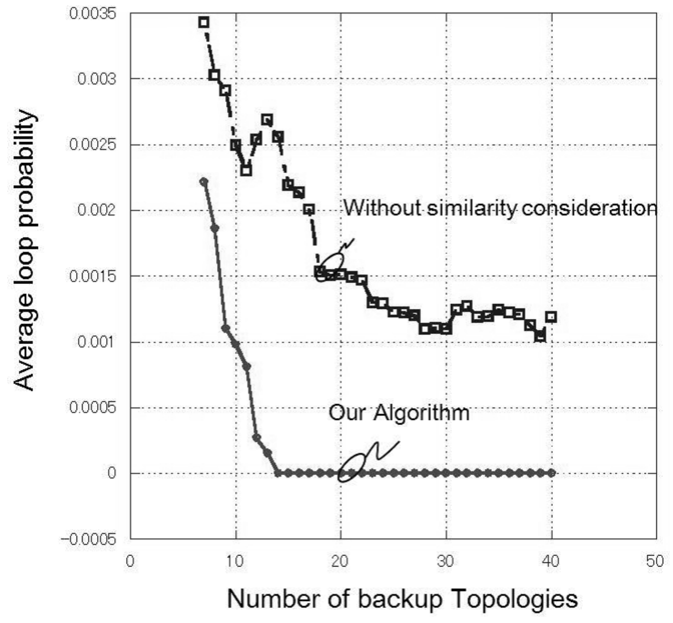


Figure 3.11: Loop probability of power-law (20 nodes 50 links).

similarity.

Our finding is that there is a correlation between the loop probability and the similarity, and to achieve loop-free forwarding on an arbitrary topology, it is essential to consider the similarity. On the cost239 model, the benchmark method also achieved loop-free forwarding, while it required about 60% more backup topologies compared to proposed (Fig. 3.9). The similarity of the benchmark method also increased although it had some randomness because cost239 model has uniform node

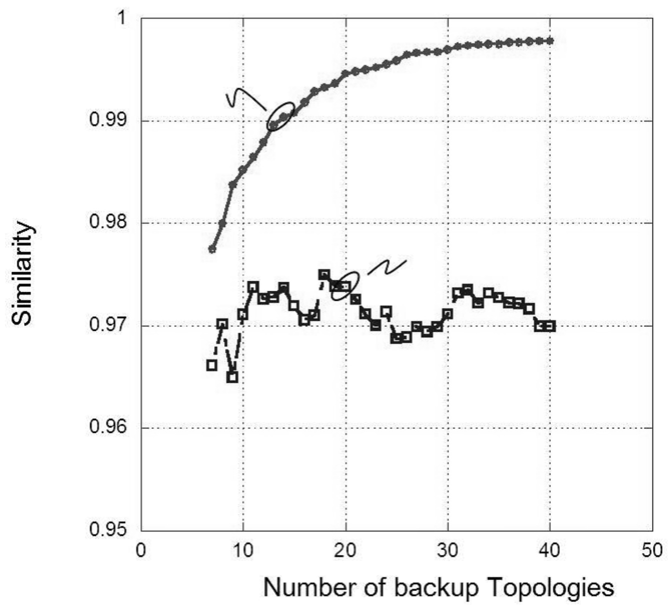


Figure 3.12: Similarity of power-law (20 nodes 50 links).

degree (Fig. 3.10). In contrast, on the power-law model, the benchmark method could not achieve loop-free forwarding even with as many as 40 backup topologies. On the contrary, the loop probability with 40 topologies increased compared to some cases with fewer topologies (Fig. 3.11). The similarity of power-law model peaks out, and it remains in the range from 0.97 to 0.975 (Fig. 3.12). From these observations, we assume that there is a correlation between the loop probability and the similarity. In addition, the benchmark method could not increase the similarity on the power-law model: our proposal, which takes the similarity with the original topology into account, can absorb the effect of variance of node degree, and has high applicability to the arbitrary form of network topologies.

3.6 Conclusion

In this chapter, we tackled the minimization of the loop probability of the IP fast rerouting on concurrent double failures. Our forwarding algorithm, which does not require explicit failure notification, can reduce the loop probability from 10^{-2} to 10^{-3} compared to existing algorithms. To achieve the objective loop probability on an arbitrary size network, our algorithm can reduce the required number of backup topologies about 35-50% compared to the loose lower bound solutions, where arbitrary two-link pairs are protected on one of the backup topologies. Because our algorithm can avoid explicit failure notification, and the increase in the number of backup topologies as network size grows is slower than existing algorithms, it has high scalability.

Chapter 4

Implementation Design of IP Fast Rerouting using OpenFlow

4.1 Introduction

This chapter presents an implementation design for IP fast rerouting. While there are many algorithms for backup topology design [24–29], implementation of fast rerouting is not very common: it requires adaptation of commercial hardware to the forwarding function. A framework to accommodate multiple routing tables was standardized by IETF [39], and some vendors implemented it [50]. However, implementation for the fast rerouting using backup topologies is still undeveloped. Though extension for software nodes [51,52] is relatively easy, modifying hardware nodes is hard and may become proprietary technology.

Recently, as one of a novel clean slate network [53], OpenFlow [54] has been proposed to provide high programmability and manageability to network. It physically separates control functions from forwarding hardware, and places them in software controllers. The forwarding table, called flow table on a forwarding hardware can be programmed through a controller: control of network does not depend on a hardware implementation.

In this chapter, we propose an autonomous IP fast rerouting method using OpenFlow. In our proposal, a dedicated controller is assigned to each forwarding element, and it locally performs restoration processes against a failure. A key point for realizing the autonomous fast rerouting is utilizing the pipeline processing with multiple flow tables defined in OpenFlow switch specification [55]. We prepare the second flow table as backup forwarding table, which stores backup routes. Before a failure occurrence, entry on a primary flow table indicates the regular output interface. If a failure is detected by the local controller, it modifies the entry of the primary flow table so that it indicates the backup forwarding table. Hot standby backup routes are preliminary stored in the backup forwarding table, and then autonomous fast rerouting is performed without global route update.

On the other hand, these preliminary stored backup routes require more memory resources. Then, we embed the backup routes compression mechanism to our IP fast rerouting method. We prepare the third flow table as a shared backup forwarding table. The routing entries on the backup forwarding table, whose destination addresses and next hop interfaces are identical, are aggregated as shared flow entries and share the same memory space on the shared backup forwarding table. Thus the size of flow table can be scaled down.

As the main contribution of this chapter is realizing IP fast rerouting by utilizing OpenFlow. By only having capability of OpenFlow, a network can realize sub-50ms autonomous fast rerouting without any extension of current forwarding hardware. In addition, we embed the flow table compression mechanism to reduce the memory consumption. It cut the table size into about half on 200 nodes network compared to the existing method [24–29]. The memory consumption generally increases as the number of backup routes increases, and the number of primary routes and backup routes are proportional to the number of flows. Then, the memory reduction by our compression mechanism increases the feasibility on actual network, which should have a large amount of flows.

The rest of this chapter is organized as follows. Section 4.2 presents our autonomous IP fast rerouting using OpenFlow, and Section 4.3 presents an algorithm for creating the shared backup forwarding table. The performance evaluation results are shown in Section 4.4. Finally, we conclude our discussion in Section 4.5.

4.2 Autonomous IP Fast Rerouting using OpenFlow

In this section, we present an IP fast rerouting using OpenFlow framework. A key idea is utilizing the pipeline processing with multiple flow tables, and switching them in accordance with the network state. Firstly, we introduce our network model, and then present the forwarding architecture for IP fast rerouting. We also present the models of restoration time with current IP [2] and our IP fast rerouting.

4.2.1 Network Model

Firstly, we describe a basic forwarding model using OpenFlow [54]. Forwarding architecture consists of OpenFlow controller and OpenFlow switches (OFSs). The controller computes the routes for OFSs, and each OFS should only forward data packets. There are two variations as the trigger for the route configuration: configuration after a flow occurrence, and preliminary configuration before a flow occurrence. We assume the latter case in this paper.

Each OFS has a forwarding table called flow table, and the flow table consists of match fields, Instructions, and Counters [55]. As one of extensions of OpenFlow 1.1 [55], a switch can have multiple flow tables. When a switch receives the packets, the switch looks up the proper flow entry whose match fields correspond to the header information of input packets. If such a flow entry exists, the switch can determine the output interface, and then forward the packets through it. Otherwise, the

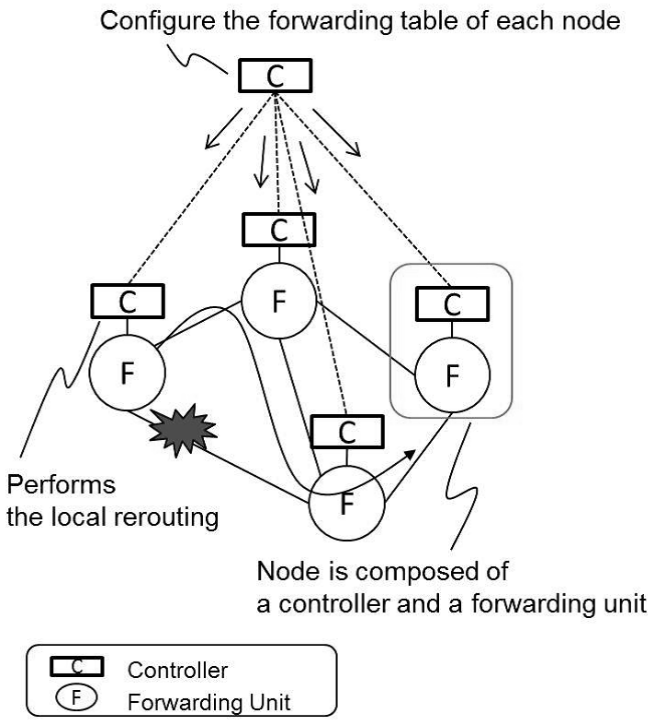


Figure 4.1: Our network model. Each forwarding hardware has a dedicated sub controller, and it performs the local control.

switch forwards the packets to the controller, or looks up the next flow table if it exists.

Our network model is illustrated in Fig. 4.1. There are two types of controllers: a central controller, which controls the whole network, and a sub controller, which controls each OFS to ignore the propagation delay. A dedicated sub controller is assigned to each OFS, and we regard the set of a sub controller and its OFS as one node (router). The central controller computes the route information, and sends it to each sub controller. Each sub controller configures the flow table of its own OFS, and performs some online processes. As software and hardware requirements of OFS, it should support OpenFlow 1.1 [55]. By the OpenFlow 1.1 specification, it supports one or more flow tables. In our network, each OFS could have three flow tables to implement our proposal as described later.

Routes are repaired by the combination of local and global repairing. Sub controllers perform simple restoration for short restoration time, and they do not depend on the state of central one. On the other hand, frequent occurrences of local restoration decrease utilization of network resources. In addition, local repair cannot handle complicated failures such as disaster. In such cases, the central controller performs global repair to optimize the entire network. In this paper, we focus on the local rerouting performed by the sub controllers and OFSs. Control by the central controller will be considered in the future.

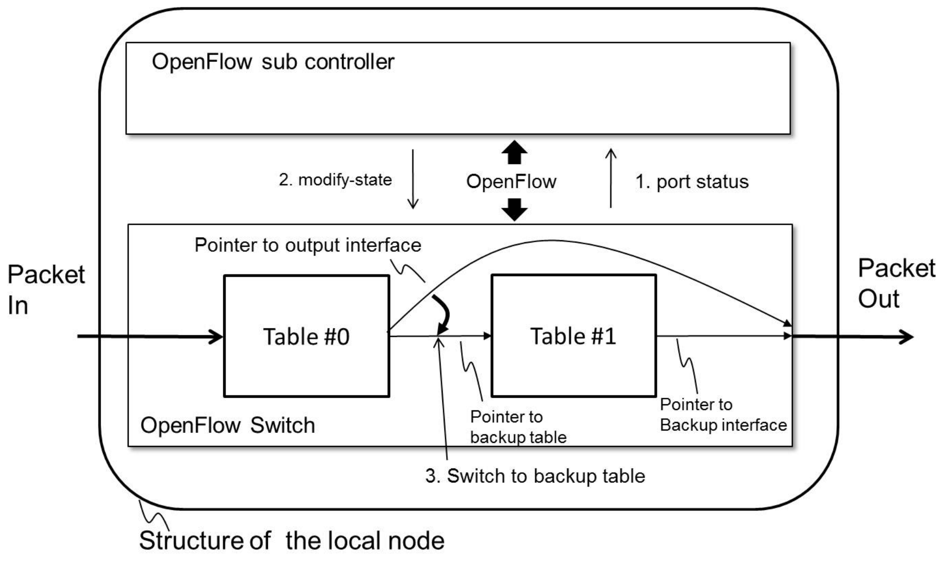


Figure 4.2: Forwarding architecture for IP fast rerouting using OpenFlow framework. Table #0 corresponds to a primary forwarding table, and table #1 corresponds to a backup forwarding table, which stores the backup routes.

4.2.2 IP Fast Rerouting using OpenFlow

This section describes the detailed mechanism of our IP fast rerouting using OpenFlow. Figure 4.2 illustrates the forwarding architecture. The interface between a sub controller and its OFS is OpenFlow. OFS has two flow tables: one is the primary forwarding table (Table #0), which contains primary routes, and another is a backup forwarding table (Table #1), which holds backup routes. In the framework of our IP fast rerouting, each node has multiple logical backup routing tables [24]. With our architecture, backup routes on the logical *backup routing tables* are stored in one *backup forwarding table*, and they are distinguished by ToS value on the match fields for simplification.

A key idea in our IP fast rerouting mechanism is utilizing the pipeline processing with multiple flow tables. Before a failure occurrence, entries in the primary flow table indicate the output interface. If a sub controller detects a failure, it modifies the corresponding entry in the primary flow table so that it indicates the backup forwarding table. Backup routes are stored in backup forwarding table in advance, and then autonomous fast rerouting is performed without global route update.

We explain the sequence of our fast rerouting using Fig. 4.3. We assume that a packet, whose destination address is 10.0.0.1, arrived at an OFS. Without failure, the packet matches the first entry of Table #0, and then it forward to interface #1. If interface #1 fails, port-status message is notified to the sub controller, and it knows the down of interface #1. Then, the sub controller updates the instructions fields of the entry on Table #0 through modify-state message. In the update, it adds the instruction for changing the ToS fields of incoming packet for avoiding the failed interface, and the instruction for looking up Table #1 instead of outgoing the

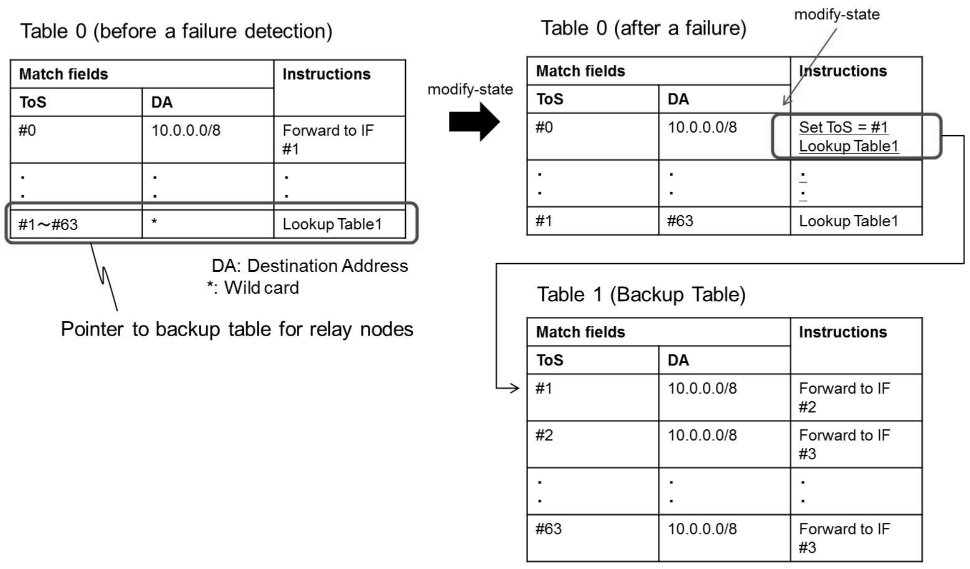


Figure 4.3: Example of flow table modification processes, and forwarding using backup forwarding table. Only one flow is illustrated for simplification.

failed interface. After the configuration by the sub controller, pipeline processing is performed for packets that match the first entry on Table #0. That is, such packets determine the output interface by referring to Table #1. On Table #1, output interface is determined by looking up it with a key composed of ToS value and destination address.

Relaying nodes, which is on the backup route, should also forward the packets using a backup forwarding table if ToS value of the packets is marked. This forwarding is realized by preparing the pointer to backup forwarding table on Table #0. In Fig. 4.3, the tail of Table #0 is the pointer to backup forwarding table. If the packets, whose ToS value is marked, arrive at relaying node, they do not match the primary entries because marked ToS does not match the primary entries. Then, they match the pointer to backup forwarding table, and are forwarded in accordance with instructions on the backup forwarding table. By preparing the pointer to the backup forwarding table, rerouting processes using the sub controller are only performed at the node that detects a failure, and rerouting processes of other relaying nodes are only performed in the OFS. Please note that OpenFlow 1.1 [55] allows the bit mask matching. By setting match fields of destination address as wild card, entries for the pointer to backup routes therefore only have to be prepared for the number of backup topologies.

While ToS value of IP header has 8 bits, OpenFlow 1.1 specified that only upper 6 bits should be used. With this restriction, maximum number of backup topologies is limited to 63. Our previous work [26], which minimized the number of backup topologies, showed that the required number of backup topologies was under 20 for 200 nodes network. The restriction is not too restrictive in this sense.

4.2.3 Discussion of Implementation Feasibility

In terms of implementation feasibility, the most important advantage of our proposal, which uses OpenFlow, compared to the one which uses commercial router is its transparency of forwarding control functions. There are two key functions to realize IP fast rerouting. One is a multi-table forwarding function, which stores multiple backup tables and forwards a packet in accordance with the proper backup table by referring to ToS value. Another is a forwarding control function at the failure detecting node, which modifies the ToS value and changes over from a primary route to a backup route.

For the multi-table forwarding function, our proposal should support OpenFlow 1.1 specification and have at least three flow tables. With the commercial router, it should support MT-OSPF [39] specification, which allows the router to have multiple routing tables. Then, implementation difficulty is roughly equal because both implementations require similar extension.

For the forwarding control function, on the other hand, programmability is completely dissimilar. Forwarding control functions of IP commercial router are coordinated to its internal unit: hardware modification is required to achieve IP fast rerouting. In addition, the forwarding control function depends on vendor's implementation, and then the extension in accordance with specific implementation is required. On the contrary, OpenFlow physically separates its control functions from the node, and they are provided as programmable area to user (e.g., carrier or service provider). As a result, our proposal realizes the user-driven and common implementation for IP fast rerouting without any extension of OFS.

Noted that we use the OpenFlow framework as a way of an IP fast rerouting implementation: we only refer to IP layer fields of incoming packets. The Openflow could provide flow-based routing scheme, which realizes granular traffic control. However, flow-based routing is beyond the IP philosophy, and may cause other study issues such as scalability and controller complexity. In this study, therefore, we only utilize one of characteristics of OpenFlow: transparency of forwarding control functions.

4.2.4 Formulation of Restoration Time with Current IP and IP Fast Rerouting

In this section, we formulate the restoration time with current IP rerouting using OSPF [2] and our IP fast rerouting architecture.

The restoration time using OSPF T_{ospf} consists of (i) failure detection, (ii) global convergence of network condition, and (iii) recomputing and updating backup routes, and it is formulated as follows.

$$T_{ospf} = T_{detect} + T_{lsao} + T_{flooding} + T_{spf-delay} + T_{spf-calc} + T_{update} \tag{4.1}$$

(i) For the failure detection, detection time (T_{detect}) was about sub-20ms in the packet over SDH/SONET (POS) network [56]. (ii) The global convergence processes consist of link state advertisements (LSAs) origination and their flooding. Let us use T_{lsao} and $T_{flooding}$ to express their time, respectively. LSA origination processes take fixed time about 12ms [57]. Before the flooding, the router that originates LSA wait 33ms [58], which is called pacing timer. Then, LSAs are notified to routers by flooding mechanism [2]. Noted that flooding processes and route recomputation processes on relaying nodes are independent, and then relaying nodes do not cause the timer delay. In a precise sense, packet transmission delays on links and routers occur in proportional to the distance between the source node and destination node. Ref. [57] reported that such delays are negligible because of the current high performance hardware and the huge-bandwidth transmission technologies. Then, $T_{flooding}$ also becomes fixed value about sub-33ms. (iii) The backup routes recomputation and updating processes consist of computation delay time $T_{spf-delay}$, backup routes computation time $T_{spf-calc}$, and forwarding table update time T_{update}. The range of $T_{spf-delay}$ is 50ms to 8000ms, and one of vendor's default value is 200ms [59]. This time is needed for the stable backup route computation against coinstantaneous LSAs advertisements. For $T_{spf-calc}$, Dijkstra algorithm is performed [2], and its time complexity is $O(N^2)$. For T_{update}, it depends on the number of entries on the forwarding table, and it takes about $146\mu s$ per entry on average [57]. Relaying routers also requires the processes for $T_{spf-delay}$, $T_{spf-calc}$, and T_{update}. These processes are performed in parallel, and then regarding the total restoration time as Eqn. 4.1 is reasonable when packet transmission delays are negligible.

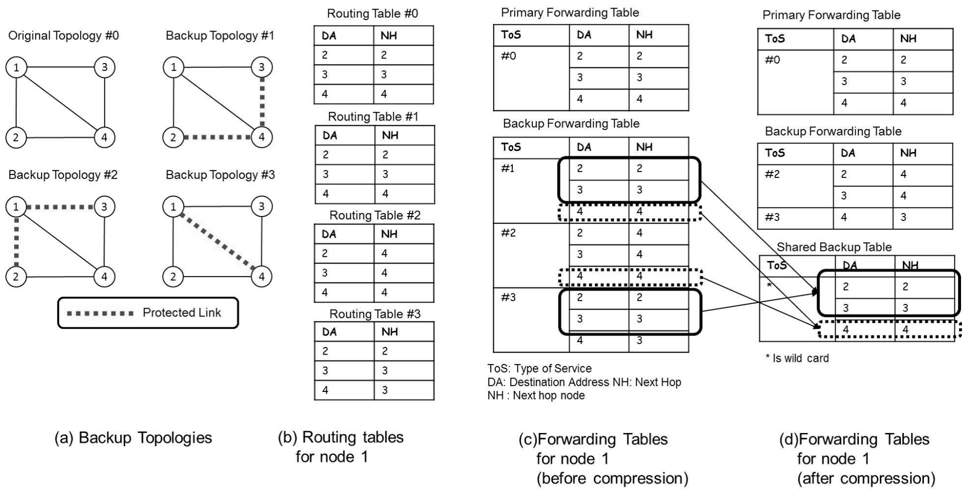
Then we formulate the restoration time of IP fast rerouting T_{frr} using above notation. IP fast rerouting process requires the failure detection and forwarding table update on a failure detecting router. There are no overheads for the restoration time on the relaying routers . Then, $T_{frr}$is formulated as follows.

$$T_{frr} = T_{detect} + T_{update} \tag{4.2}$$

There are two types of forwarding table update: the full update and the partial update. The partial update only updates the entries which are impacted by a failure while the full update updates all entries. While the table update becomes partial update with our architecture, with the OSPF rerouting, it is implementation matter if partial update is supported or not [57]. When the size of forwarding table of router R, the total number of flows, and the number of flows which are included on failed link l, are denoted as $T(R)$, F_{all}, and $F(l)$, the number of entries, which are updated in the partial update environment, is denoted as $T(R) \times (F(l)/F_{all})$. That is, the number of updated entries is proportional to the ratio of the failed flows.

4.3 Flow Table Compression with Shared Flow Entries

As mentioned in Sec. 4.2, fundamental IP fast rerouting is realized by utilizing OpenFlow. On the other hand, this method increases the memory consumption. In this section, we embed the compression mechanism to our IP fast rerouting by using shared backup forwarding tables. Firstly we provide the algorithm for shared backup table creation, and then provide forwarding architecture using it.


Original Topology #0
Backup Topology #1
Backup Topology #2
Backup Topology #3
Protected Link
(a) Backup Topologies
(b) Routing tables for node 1
(c)Forwarding Tables for node 1 (before compression)
(d)Forwarding Tables for node 1 (after compression)
* Is wild card

Routing Table #0

DA	NH
2	2
3	3
4	4

Routing Table #1

DA	NH
2	2
3	3
4	4

Routing Table #2

DA	NH
2	4
3	4
4	4

Routing Table #3

DA	NH
2	2
3	3
4	3

Primary Forwarding Table

ToS	DA	NH
#0	2	2
	3	3
	4	4

Backup Forwarding Table

ToS	DA	NH
#1	2	2
	3	3
	4	4
#2	2	4
	3	4
	4	4
#3	2	2
	3	3
	4	3

ToS: Type of Service
DA: Destination Address NH: Next Hop
NH : Next hop node

Primary Forwarding Table

ToS	DA	NH
#0	2	2
	3	3
	4	4

Backup Forwarding Table

ToS	DA	NH
#2	2	4
	3	4
#3	4	3

Shared Backup Table

ToS	DA	NH
*	2	2
	3	3
	4	4

Figure 4.4: Relationship between backup topologies and backup routing tables, and an overview of flow table compression.

4.3.1 Shared Backup Forwarding Table Creation Algorithm

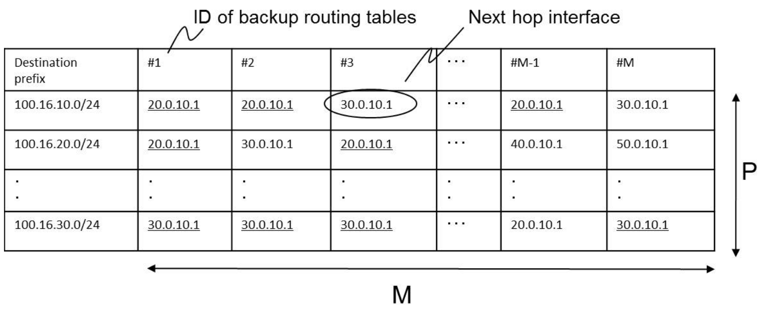
In existing study [24–29], backup topology and backup routing table match one-to-one as shown in Fig. 4.4 (a) and (b). The number of flow entries therefore increases in proportion to the number of backup topologies. On the other hand, while backup topologies are mutually unique, flow entries for certain nodes in backup routing tables are not necessarily unique. That is, redundant entries exist. Figure 4.4 shows an example. Flow entries whose destination addresses are 2 or 3 in backup routing table #1 corresponds to entries in backup routing table #3. Flow entry whose destination address is 4 in backup routing table #1 corresponds to the entry in backup routing table #2. Therefore, we can aggregate them as a shared flow entry, and store the shared flow entries to one common backup forwarding table called shared backup forwarding table.

Key point of shared backup forwarding table creation algorithm is registering the only flow entries with the maximum number of sharing, if there are multiple candidates. That is, we prohibit duplicated entries with the same destination address in the shared backup forwarding table. If we allow the duplicative registration, there are no effect of sharing because each entry on shared backup forwarding table should be distinguished the key composed of forwarding table ID and destination address. Here is an example. The set of routing table ID, destination address and next hop interface is denoted as (rID, DA, NH). We assumed that there are backup routing tables #1, #2, which have the same destination address $DA1$ and next hop interface $NH1$, and there are other backup routing tables #3, #4, #5, which have the same destination address $DA2$ and next hop interface $NH2$. If we register the above entries to the shared backup forwarding table, the number of entries becomes five: *(#1, DA1, NH1), (#2, DA1, NH1), (#3, DA2, NH2), (#4, DA2, NH2), (#5, DA2, NH2).* Then, if we only register them for #3, #4, #5, it becomes one because wild card is allowed: $(any, DA2, NH2)$.

Figure 4.5 illustrates our algorithm. The input to the algorithm is all backup

```
1: create matrix in_addr_t nextHop[P][M];
2: create vector int sharedNum[sizeof(in_addr_t)];
3: for (int p = 0; p < P; p++)
4:     for (int m = 0; m < M; m++)
5:         sharedNum[nextHop[p][m]]++;
6:     maxValue = Search(sharedNum);
7:     register "(Dst, NH) = (p, maxValue)" to Shared tbl;
8:     remove "(Dst, NH) = (p, maxValue)" from backup tbls;
9:     init (sharedNum);
10: end
```

Figure 4.5: Shared backup table creation algorithm.



Destination prefix	#1	#2	#3	⋯	#M-1	#M
100.16.10.0/24	20.0.10.1	20.0.10.1	30.0.10.1	⋯	20.0.10.1	30.0.10.1
100.16.20.0/24	20.0.10.1	30.0.10.1	20.0.10.1	⋯	40.0.10.1	50.0.10.1
:	:	:	:	:	:	:
100.16.30.0/24	30.0.10.1	30.0.10.1	30.0.10.1	⋯	20.0.10.1	30.0.10.1

Figure 4.6: Structure of matrix nextHop[P][M].

routing tables, and its output is updated backup routing tables and a shared backup forwarding table. Firstly, we create the nextHop matrix (line 1), and it is illustrated in Fig. 4.6. The number of sharing is stored in sharedNum vector, and it is initialized in line 2. Then, following processes are repeatedly performed for each destination prefix on nextHop matrix (line 3-9). The number of sharings is counted for destination prefix p, and it is stored in sharedNum vector (line 4-5). For example, sharedNum["20.0.10.1"] is three, and sharedNum["30.0.10.1"] is two for destination prefix 100.16.10.0/24 in Fig. 4.6. Then, our algorithm selects the next hop interface, whose shared number is the maximum, and save it to maxValue variable (line 6). With this example, "20.0.10.1" is stored in maxValue for the destination prefix 100.16.10.0/24. Then, the set of destination prefix p and next hop interface maxValue is registered in shared backup forwarding table (line 7), and removed from backup routing tables, which has the corresponding entry (line 8). Finally, sharedNum vector is initialized (line 9), and the algorithm continues above processes for next destination prefixes. When all destination prefixes are scanned, our algorithm finishes (line 10). For example in Fig. 4.6, underlined next hop interfaces are registered in the shared backup forwarding table.

Finally, we discuss the computation of our algorithm. The computation time of processes on line 4-5, and line 8 becomes $O(M)$. The search processes on line 6 depends on the size of address type (e.g., 32 bits for IP address), and does not

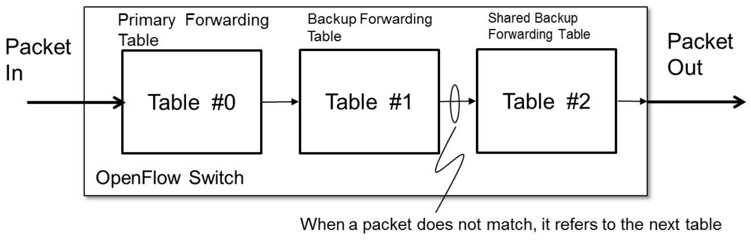


Figure 4.7: Forwarding sequence using shared backup table.

depend on the size of input information. A process on line 7 does not depend on it. Above processes are performed P times (line 3), and then computation complexity becomes $O(PM)$.

4.3.2 Forwarding Architecture using Shared Backup Forwarding Table

Packet forwarding using the shared backup forwarding table is also realized using pipeline processing. We prepare the flow table #2 as a shared backup forwarding table (Fig. 4.7). Entries aggregated to the shared backup forwarding table do not appear in the backup forwarding table (Table #1). Therefore, the result of looking up such entries in Table #1 is unmatched. OpenFlow 1.1 has the function that unmatched packet causes look up of the next flow table. The unmatched packet lookups at Table #1 results in looking up at the shared backup forwarding table. In the shared backup forwarding table, we set the match fields of ToS value as wild card, and then output interface is determined by only using destination address fields.

4.4 Performance Evaluation

The goal of our evaluation is showing validity of our IP fast rerouting which uses OpenFlow framework. The main requirement of IP fast rerouting is to achieve sub-50ms recovery [24]. Therefore, we evaluate whether our proposal, which needs the online table updating, can achieve sub-50ms recovery. In addition, we also compare our proposal to the current IP rerouting (OSPF) [2] for showing the superiority of IP fast rerouting itself. This is because that existing works [24–29] focused on the backup topology design algorithm, and there were no detailed implementation and evaluations in terms of restoration time.

As previously described, main overhead of our IP fast rerouting is increase of memory consumption caused by the preliminary stored backup routes. We also clarify this overhead compared to OSPF [2].

4.4.1 Simulation Conditions

Firstly, we show the effectiveness in terms of the restoration time of IP fast rerouting compared to OSPF [2] using our formulation described in Sec. 4.2. For the variables $T_{spf-calc}$ and T_{update}, we measure them by computer simulation. $T_{spf-calc}$ is equal to the Dijkstra shortest-path calculation time, whose computation increases as the square of the number of nodes. For the T_{update}, we count the number of entries in the forwarding table for each router, and it is multiplied by $146\mu s$. For the partial update, we compute the worst case where the link, which has the most number of flows, fails. Above simulations are performed in the system with CentOS 5.5, quad-core Xeon 1.86GHz, and 32GByte memories. For the fixed value T_{detect}, T_{lsao}, $T_{flooding}$, and $T_{spf-delay}$ are set to fixed values 20ms, 12ms, 33ms and 200ms, respectively.

For the overhead evaluation, we evaluate our IP fast rerouting compared to the existing IP fast rerouting methods [24–29] and OSPF [2]. Note that existing IP fast rerouting methods [24–29] assumed that the number of backup forwarding table was proportional to the number of backup topologies such as Fig. 4.4 (c). As an indicator of the effectiveness, we use the total number of flow entries on primary, backup, and shared backup forwarding tables. Note that there are multiple nodes on a network, and then we compute the average number of total flow entries among all the nodes.

As the destination addresses in the forwarding tables, the routers and their interfaces in the system are assumed. Also we assume that no routes aggregation is performed. Network topology is power-law model [43]. In the power-law model, most nodes have a small number of links, while a small portion of nodes have a large number of links. This model is often used as a representation of the actual Internet [43]. The number of nodes is varied from 20 to 200 nodes, and node-degree is set to two: each node has four links on average.

4.4.2 Restoration Time

Figure 4.8 shows the restoration time with OSPF and IP fast rerouting when the number of nodes changes. Our IP fast rerouting can achieve sub-50ms restoration on our conditions. Though our IP fast rerouting architecture requires the partial update process for the restoration, it is negligible. Therefore, even if we deploy an alternative IP fast rerouting method, which can change over to backup routes as node-internal processes, restoration time is assumed to be almost same. In this evaluation, we only consider the internal routes for routers and interfaces addresses. Even if the external routes exist, we can curb the increase of table update time by using hierarchical forwarding information base (FIB) technology [60].

From these results, though the OSPF cannot achieve sub-50ms restoration, it achieves sub-second restoration and it seems to be faster than expected. Noted that this restoration time is the optimal situation. If all LSAs caused by a single failure cannot be received within the computation delay (200ms), computation hold time for backup routes computation results, and it is 5sec by default [59]. These gaps in the expected speed of LSAs are caused by the sequential failures, and difference of router equipment or the failure detection technologies. Therefore, OSPF even allows the longer computation delay (e.g., 8sec). Then, the finding of this paper is that computation delay for the stable global convergence is main factor of restoration time, and dynamic processes such as Dijkstra computation

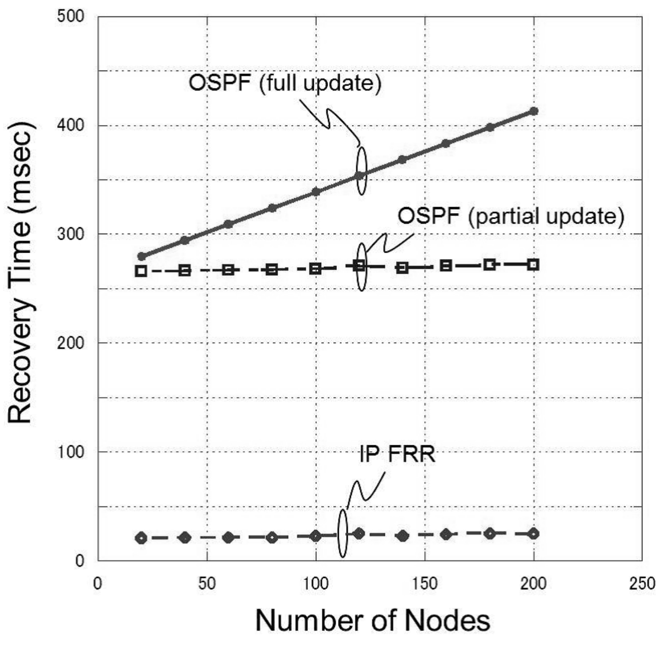


Figure 4.8: Restoration Time with optimal OSPF(full/partial), and IP fast rerouting.

and partial update are negligible on the practical network size [2]. On the other hand, our IP fast rerouting, which performs local rerouting, does not require the global advertisement. Then it always achieves the sub-50ms restoration.

4.4.3 Compression Effect

Figure 4.9 illustrates results for the total number of flow entries. Though the increase of the total number of flow entries with IP fast rerouting methods is inevitable, our IP fast rerouting, which compress the backup forwarding table, mitigates the impact of increase. With the existing method, the additional number of flow entries is approximately the number of backup topologies times the number of flow entries for OSPF. For example, it is 6 times the number of flow entries for OSPF on 20-node network. By using our compression algorithm, the reduction compared to the existing method is about 30% on 20-node networks. In addition, the reduction is higher for larger networks (e.g., 45% reduction on 200-node networks).

The reason reduction effect increases as the number of nodes increases is as follows. If network becomes large, the number of links that should be protected increases, and then the required number of backup topologies also increases. Because the number of backup routing tables increases in proportion to the number of backup topologies, the number of flow entries before compression increases. On the other hand, the frequency of sharing also stochastically increases as the number of backup routing tables increases. This is more noticeable for the nodes whose node-degree is low because its available links is limited, and then a probability to use same link increases. In addition, in large networks, backup routing tables,

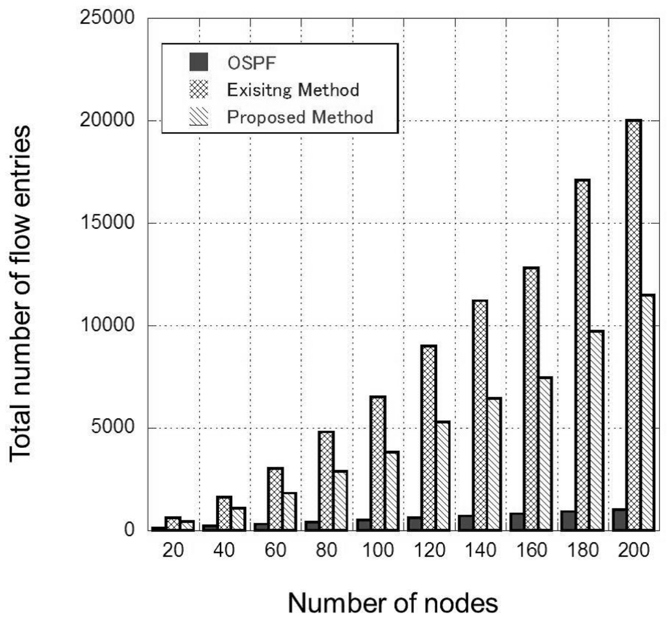


Figure 4.9: Effectiveness of compression algorithm on Power-law network.

which have redundant entries, tend to be made. For example, even if the property (protected or not) of links with node A, which is separately-located with node B, changes, it does not affect the backup routing tables of node B. Such situation is possibly to occur as network becomes large. Therefore, compression effect increases as network become large. By contrast, increase of reduction effect is gradual when the number of nodes increases. We suppose this is because of prohibition of duplicated registration for the same destination addresses in the shared backup forwarding table. The number of entries that is identical each other increases as the number of nodes increases because of above reasons. However, there is no guarantee that each entry has the same next hop interface, and then reduction effect becomes gradual.

Finally, we discuss the effectiveness of compression when each node has external user routes. If each node uniformly has the external user routes, the same results as shown in Fig. 4.9 are expected: external user routes, which belong to the sharable destination prefix, are also sharable. Then, we assume that external user routes on each node are unevenly-distributed. As actual model, the nodes, whose node-degree is high, are commonly central nodes, and they have a large number of external user routes. Therefore, we assume that the number of external user routes is proportional to the node-degree of each node. With this model, there is inversely proportional relationship between the number of links (node-degree) at a node and the registered entries on the node's flow tables. For example, the nodes, whose node-degree is low, stochastically connect to the nodes, which have a large number of links. Then the number of registered flow entries for such nodes increase. As previously mentioned, for the nodes whose node-degree is low, sharing effect is high. That is, the nodes, whose node-degree is low, have a large number of flow entries but their compression effect becomes high. On the contrary, the nodes, whose node-degree is high, have a low number of flow entries but their compression effect becomes low. As a result, compression effect is proportional to the size of flow tables, and then compression effect close to results on Fig. 4.9 is expected for the unevenly-distributed external user routes.

4.5 Conclusion

We proposed the autonomous IP fast rerouting method using OpenFlow. By utilizing OpenFlow, we can achieve sub-50ms restoration without any extension of current forwarding hardware. In addition, our flow table compression algorithm can reduce the size of flow table about 45% compared to existing algorithms. This compression effect also increases the feasibility of IP fast rerouting on actual networks.

Chapter 5

Relaxed Maintenance Network using Dynamic 1+1 Path Protection

5.1 Introduction

As an alternative to IP, MPLS [3] technology, which provides connection-oriented paths, was also proposed for backbone network control. This technology enables network operators to establish explicit paths among nodes, and a network is operated in accordance with its operator ’ s proactive network design. MPLS provides fast restoration against a failure by 1+1 path protection mechanism [5]. The 1+1 path protection provides primary and backup paths, which are link- or node-disjoint paths. Traffic data is transmitted on both paths, and the receiver node switches receiving data: it can achieve lossless data transmission.

In terms of operation and maintenance, on the other hand, if the primary or backup path becomes unavailable because of failure or planned maintenance, the network operator is required to quickly repair the failed component to maintain the availability specified in the customer contracts. Repairing physical failures such as fiber-cut [61] sometimes requires a long time which includes equipment procurement. In addition, failure occurrence at night incurs extra cost on human resources.

In this chapter, we propose the network architecture and path computation algorithm to maintain 1+1 path protection even after a single failure by dynamically assigning a new backup path to allow long repair time. Establishing a new backup path in a few seconds minimizes the violation of the availability specified customer contracts, and prolongs the allowable maintenance time.

We evaluated our proposal in two perspectives: network design problem and network sustainability problem. From the network design perspective, we evaluated total equipment cost, path length, and blocking probability of our proposed dynamic approach and a static design approach on various forms of networks. The static design approach we used is the one which establishes three independent paths in advance. We showed that our dynamic path protection could maximally avoid the influence of sparse-node existence. From the network sustainability perspective, we showed that our algorithm could reduce blocking probability and allowed long maintenance time for repairing compared to simple benchmark algorithms.

The rest of this chapter is organized as follows. In Section 5.2, we introduce related works and our problem statement. Section 5.3 presents our network architecture and path computation algorithm. Section 5.4 provides formulation of maintenance time based on blocking probability. Our evaluation results are shown in Section 5.5. Finally, we conclude our discussion in Section 5.6.

5.2 Related Works and Problem Statement

5.2.1 Related Works

Path protection technologies are classified into 2 classes: 1+1 path protection where traffic data is transmitted on both paths [5], and 1:1 path protection where primary and backup paths are established but traffic data is transmitted only on primary path. The 1:1 path protection allows shared path protection where bandwidth for backup path is shared among different source and destination pairs [62]. This bandwidth share is effective in terms of the total equipment cost reduction. On the other hand, reliability degrades compared to the 1+1 path protection because online switching from the primary path to backup path is required when failure occurs. In addition, if two paths, which share the same links as their backup paths, simultaneously fail, one path cannot be recovered.

The basic policy of 1+1 path protection [5] is that the source node is allowed to split the traffic, and the destination node is allowed to switch between traffics. The intermediate nodes are restricted to only transmitting the traffic. Primary and backup paths are computed as disjoint paths, which do not share the same links or nodes. Disjoint paths are computed by suurballe method [63] in polynomial time. The suurballe method can also produce N-disjoint paths if a network topology has N-edge connectivity.

Recently, as generalized concept of 1+1 path protection, integer generalized dedicated protection (IGDP) problem [64, 65] and multi path protection (MPP) problem [66] were studied. In IGDP problem, arbitrary nodes are allowed to split the traffic or switch between traffics. The number of backup routes increases as split points increases. Then, it can handle multiple failures such as shared risk link group (SRLG) failures though required bandwidth increases. The IGDP problem belongs to NP complete problem, and it can be formulated as an integer linear programming (ILP) [65]. In MPP problem [66], integer constraint of IGDP problem is relaxed to real number constraint, and then it is formulated as a linear programming.

5.2.2 Problem Statement

The 1+1 path protection is a promising technology to realize reliable networks. In terms of network operation, however, rapid repairing of failed equipment is

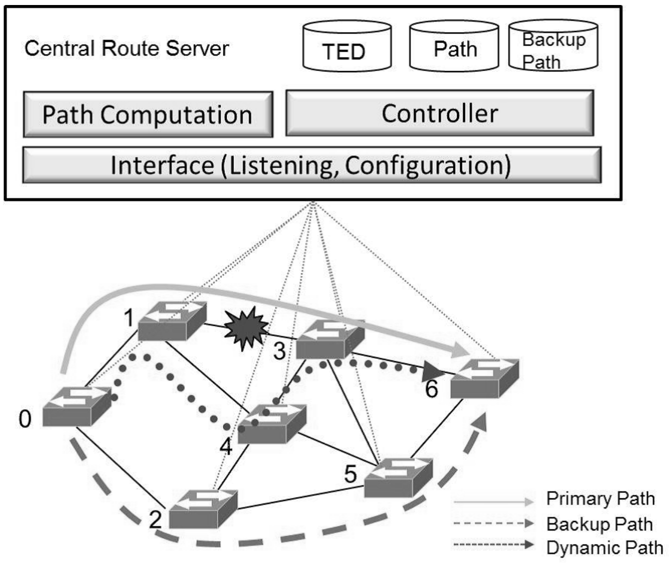


Figure 5.1: Dynamic path reconfiguration architecture.

also important. As previously described, there are many studies for 1+1 path protection [64–66]. These studies mainly focused on path-design algorithm, and there were few discussions for the maintenance time.

From the above observation, our problem is establishing network architecture to maintain 1+1 path protection as much as possible and as a consequence relaxing of the restriction in terms of the maintenance time.

5.3 Network Architecture and Algorithm for Dynamic 1+1 Path Protection

For maintaining 1+1 path protection even on a failure, an alternative backup path should be rapidly prepared after the failure occurrence. In this section, we present the network architecture, path computation algorithm, and bandwidth calculation method for our dynamic path reconfiguration system.

5.3.1 Network Architecture

Our dynamic path reconfiguration architecture consists of a network and a central route server (Fig. 5.1). A central route server has control interface which collects network state and configures paths, control function, and route computation function. Each node on the network connects with the central route server through the control interface. When the central route server detects a failure, it configures backup routes and transmits their information to nodes on the routes, and alternative backup paths are dynamically established. We illustrate an example to establish a backup path between node 0 and 6 using Fig. 5.1. When the link 1-3 fails and the current primary path 0-1-3-6 becomes down, the backup path 0-2-5-6 behaves as a new primary path. At this time, failure information is also notified to the central route server, and it configures a new backup path 0-1-4-3-6, which does not includes the failed link and links on new primary paths (0-2-5-6): 1+1 path protection state is immediately recovered. Hereafter, we describe the basic architecture and the implementation method of the dynamic path reconfiguration in detail.

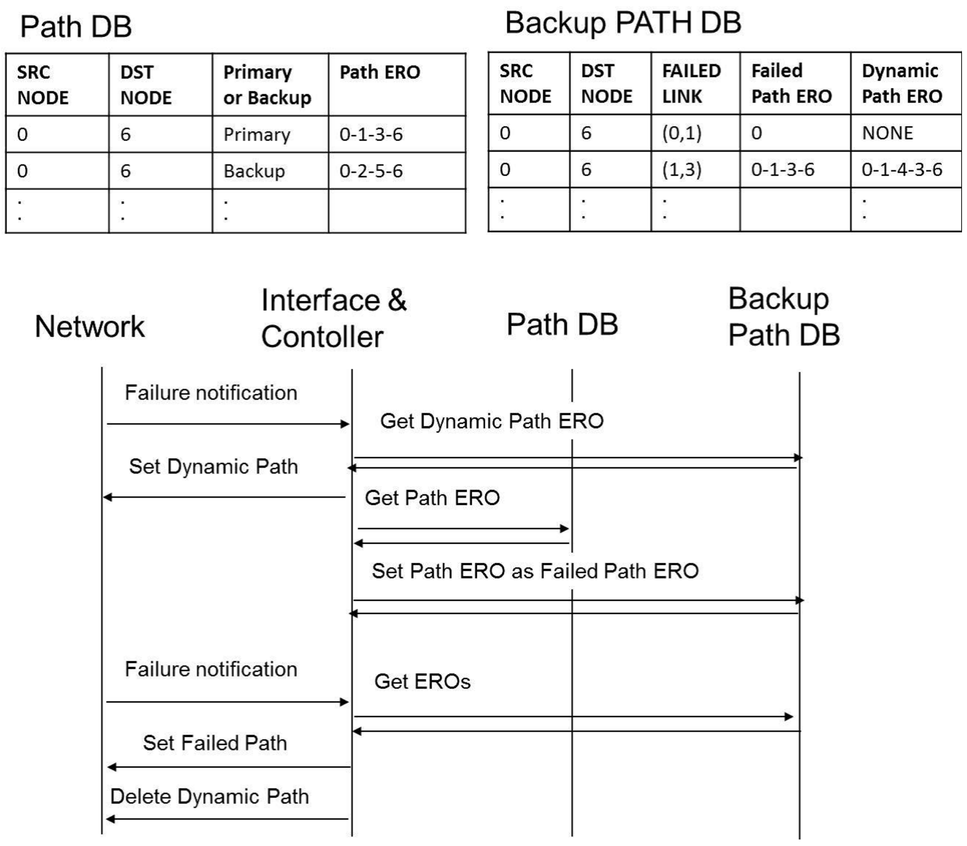
Basic Design

Before the operation of the dynamic protection, the network topology, which includes the information of nodes, links and their bandwidths, and paths are stored to the databases of central route server. In this chapter, the database of network topology is described as traffic engineering database (TED) [67]. In the case that the network is already operated as a 1+1 path protected network, utilization of links is also stored to the central route server. In the route computation functions, dynamic protection paths are computed based on TED by assuming single failure. The route computation algorithm is described in 5.3.2 later. One dynamic protection path consists of a source node, a destination node, a failed link or node, an explicit route object (ERO) of failed path and ERO of dynamic path. ERO means the route of a path represented by a sequence of node IDs [68]. ERO of failed path is used for storing failed path information, and its value is initialized as empty. Then, this dynamic path information against possible single failure is stored to backup path database. Primary and backup path information is also stored in path database. The columns of path database consists of source node, destination node, identifier of path (primary or backup), and ERO of the path (Fig. 5.2).

When network state changes in the operation phase with our dynamic protection, the central route server detects network change and dynamically configures new paths. Figure 5.2 illustrates the sequence for the recovery process on failure and for the reversion process on repair. When the control interface detects a failure, the control function searches a proper backup path entry from the backup path database with a failed link or node as the key. Then, the path specified by the dynamic path ERO in a matched entry is configured through the control interface. Then, a failed path ERO is also searched from the path database with a key composed of source node, destination node, and failed link or node on the backup path entry. The EROs of faulty paths in the path database are registered as failed path EROs. When the control interface detects repairing of failed equipment, the control function searches the reversioning path entry from the backup path database with repairing equipment as the key. The failed path ERO in the backup path database corresponds to its reversioning path. Then, the path for reversion is established, and the former established dynamic path is removed through the control interface.

Implementation Method

Recently, centralized control approaches such as path computation element (PCE) for Generalized-MPLS (GMPLS) network [70] and OpenFlow protocol [54] has appeared. In this section, we introduce two types of implementations for dynamic



Path DB

SRC NODE	DST NODE	Primary or Backup	Path ERO
0	6	Primary	0-1-3-6
0	6	Backup	0-2-5-6
:	:	:	

Backup PATH DB

SRC NODE	DST NODE	FAILED LINK	Failed Path ERO	Dynamic Path ERO
0	6	(0,1)	0	NONE
0	6	(1,3)	0-1-3-6	0-1-4-3-6
:	:	:		:

Figure 5.2: Control sequence of dynamic path reconfiguration and reversion.

path reconfiguration: control plane-based and management plane-based.

In the control plane-based approach, OSPF-TE protocol [67] can be used for collecting network state as TED: it could collect the relation of connection, bandwidth of links, and utilization ratio of links. The failed point is detected by comparing the initial TED and the latest TED. For the path configuration, signaling protocol like RSVP-TE [68] can be used. In this situation, the central route server notifies the proper ERO to the source node of the path thorough path computation element protocol (PCEP) [69]. Then, ERO is transmitted along to the path by the signaling protocol.

In the management plane-based approach, the central route server directly configures forwarding table called forwarding information base (FIB) of each node. As ways of direct configuration of FIB, ForCES [71] and OpenFlow [54] was proposed. Link layer discovery protocol (LLDP) can be utilized for collecting TED [72]. OpenFlow protocol is also used for detecting failed point or collecting traffic information.

5.3.2 Algorithm for Dynamic Disjoint Path Discovery

In this section, we propose weighted-constrained shortest path first (W-CSPF) algorithm, which utilizes available resources to establish efficient dynamic paths. With the current constrained shortest path first (CSPF) algorithm [73], the links,

Table 5.1: Notation for W-CSPF Algorithm.

$G = (V, E)$	Graph with nodes V and undirected links $E, e_{i,j} \in E$
w_{ij}	weight of link $e_{i,j}$
D^k	traffic demand of flow k
$B_{i,j}$	available bandwidth of link $e_{i,j}$
$X^k_{i,j}$	1 if working path k use link $e_{i,j}$, otherwise 0
$Y^k_{i,j}$	1 if backup path k use link $e_{i,j}$, otherwise 0
$s(k)$	source node of flow k
$d(k)$	destination node of flow k
$SPT(s, d, G)$	shortest path from s to d on graph G

which do not satisfy the bandwidth request of paths, is removed from the original topology. Then, shortest paths are computed based on the filtered topology. As long as the filtered topology has connectivity, path between an arbitrary source and destination node is established. On the other hand, link utilization on the filtered topology depends on the shortest path first (SPF) algorithm, and then provided paths by CSPF has potential to result in inefficient link utilization. To be more precise, we assume the situation of dynamic path configuration when a link failure occurs. Because each link ordinarily accommodates multiple primary or backup paths, multiple dynamic paths should be configured against a single failure. If we use the CSPF algorithm for dynamic path computation, bias of link utilization and congestion may occur due to above SPF mechanism: as paths are sequentially configured, links, which do not have enough bandwidth, increase. As a result, connectivity between an arbitrary source and destination node could be lost. Compared to the CSPF algorithm, our W-CSPF algorithm not only filters out the unsatisfied links, but also considers the link utilization of links on the filtered topology: it weights links in accordance with their available resources.

Inputs to our algorithm are a physical topology and a traffic matrix. Traffic matrix expresses the amount of data traffic between every pair of network ingress/egrress points [74]. For example, the size of a traffic matrix is $|V|^2$ when the number of nodes is denoted as $|V|$. The primary and backup paths are pre-computed using the Suurballe method [63], which computes two link-disjoint paths between source and destination nodes. We assume link failure occurs on the primary or backup path and compute a dynamic disjoint path as the new backup path. The details of our algorithm are shown in Table 5.2, using the notations listed in Table 5.1.

In a directed graph, $G = (V, E)$, we assume link $e_{ab} \in E$ failed. Then we apply the following processes to flow k (primary or backup path) included in link e_{ab}. If flow k is the primary path, a set of links $E1$ is created (line 2,3). $E1$ is a set of links that compose a backup path. If flow k is the backup path, a set of links $E1$ is created (line 4,5). In this instance, $E1$ is a set of links that compose a primary path. Then, a set of links $E2$ is created (line 6). $E2$ contains links whose available

Table 5.2: W-CSPF Algorithm.

```
1   for all $k$ $\{k \mid X^k_{ab} = 1 \mid\mid Y^k_{ab} = 1\}$ do
2       if $(X^k_{ab} = 1)$ then
3           $E1 := \{e_{i,j} | Y^k_{ij} = 1\} \quad i, j \in V$
4       if $(Y^k_{ab} = 1)$ then
5           $E1 := \{e_{i,j} | X^k_{ij} = 1\} \quad i, j \in V$
6       $E2 := \{e_{i,j} | B_{i,j} < D^k\} \quad i, j \in V$
7       $E3 := E \backslash \{E1 \cup E2 \cup e_{a,b}\}$
8       $forall$ $e_{i,j} \in E3$ do
9           $w_{i,j} := int\{\max_{e_{l,m} \in E3}(B_{l,m})/B_{i,j}\}$
10      $G' := (V, E3)$
11      return $SPT(s(k), d(k), G')$
12  end
```

bandwidth do not satisfy the required bandwidth of flow k in graph G. Then, a new set of links $E3$ is created(line 7). $E3$ is composed of E, except for $E1$, $E2$, and failed link e_{ab}. At this point, a set of links $E3$ that is available for accommodating flow k is prepared. Then, link weight that is inversely proportional to available resources is assigned to each link in $E3$ (line 8,9). Dynamic path are computed as minimum cost path based on link weights. Therefore, by this process, the links with enough available resources are preferentially used for flow accommodation. Finally, the shortest path from $s(k)$ to $d(k)$ is computed on graph G'. As well, simple shortest paths (SPF) are computed by skipping the processes in line 6, 8, and 9, and CSPF paths are computed by skipping the processes in line 8 and 9.

Our approach can also be applied to node failure. The primary and the backup paths are computed also using the Suurballe method, which computes two node-disjoint paths between source and destination nodes. Then, we slightly extend our dynamic disjoint path discovery algorithm. When we denote failed node f, failed links becomes $e_{if}, i \in V$. Then we use our algorithm for flows $X^k_{if}, i \in V$ and $Y^k_{if}, i \in V$ against link failures .

As for the computation time, shortest path computation (line 11) is dominant, and computation time of the shortest path calculation is $O(|V|^2)$; therefore, the computation of our algorithm is $O(|k||V|^2)$ with the number of flows $|k|$ on link e_{ab}.

5.3.3 Bandwidth Design for Dynamic Disjoint Path Discovery

If we know the traffic matrix, we can compute the reserved bandwidth, which are required resources for links. We propose a reserved bandwidth-computation method for reliable network design. First the volume of traffic flows that occur at all source and destination node pairs, is allocated to each link of the primary and backup paths. At this point, the reserved bandwidth for the primary and backup paths is computed.

Next, we compute the reserved bandwidth of the dynamic path $Bw(e), e \in E$. First, dynamic paths are computed by assuming link i failure. Then, the volume of traffic flows that are required by the dynamic paths is allocated to each link $e \in E$ as reserved bandwidth $Bw(e, i)$. We perform the path computation algorithm against each link failure. At this time, the required bandwidth of each link $Bw(e, i)$, $e, i \in E$ is independent in each pattern of link failure i. Therefore, reserved bandwidth can be shared. That is, only the maximum value of required bandwidth $max_{i \in E} Bw(e, i)$ is allocated as the reserved bandwidth $Bw(e)$. Then, we avoid unnecessary bandwidth allocation by sharing the reserved bandwidth.

5.4 Formulation for Maintenance Time

In this section, we formulate maintenance time. Maintenance time means that which could be assigned for repairing of failed components. A network is denoted as a directed graph $G = (V, E)$ with nodes V and links E. The set of flows (source and destination pairs) that are included in link $x \in E$ is denoted as F_x. For flow $f \in F_x$, the set of links that compose primary path, backup path, and dynamic path are denoted as $E_P(f)$, $E_B(f)$, and $E_D(f)$ respectively. The dynamic path does not always exist for flow f: if the minimum cut between source node and destination node is less than three, we cannot recover the link failure that is composed of the minimum cut. Therefore, we define the blocking probability β_x that are fraction of flows that have no own dynamic path, and are included in link x. The mean time between failure (MTBF) of link e is denoted as $MTBF(e)$, and the mean time to repair (MTTR) is denoted as $MTTR(e)$. For simplification, $MTTR(e)$ is assumed to be the same value among all links. Availability of 1+1 path protection after the link x failure is denoted as $A_{1+1}(x)$ and availability with our proposal is denoted as $A_d(x)$.

Availability $\alpha(e)$ of link $e \in E$ is denoted as follows.

$$\alpha(e) = \frac{MTBF(e)}{MTBF(e) + MTTR(e)} \tag{5.1}$$

Using Eq. 5.1, availabilities of primary path A_P, backup path A_B, and dynamic path A_D can be defined as follows;

$$A_P = \prod_{e \in E_P(f)} \alpha(e) \tag{5.2}$$

$$A_B = \prod_{e \in E_B(f)} \alpha(e) \tag{5.3}$$

$$A_D = \prod_{e \in E_D(f)} \alpha(e) \tag{5.4}$$

The 1+1 path protection has two redundant paths, and our dynamic architecture on an ideal network topology (e.g. 3-edge connected graph) has three redundant paths; hence, the availability of 1+1 path protection $A_{1+1}(x)$ and availability with our proposal on an ideal environment $A_d^{ideal}(x)$ are given as follows;

$$A_{1+1}(x) = \left(\sum_{f \in F_x} \{1 - (1 - A_P)(1 - A_B)\} \right) / |F_x| \tag{5.5}$$

$$A_d^{ideal}(x) = \left(\sum_{f \in F_x} \{1 - (1 - A_P)(1 - A_B)(1 - A_D)\} \right) / |F_x| \tag{5.6}$$

Networks do not necessarily have three disjoint path. Availability $A_d(x)$ can be defind as follows using Eq. 5.5 and Eq. 5.5 with blocking probability β_x.

$$A_d(x) = (1 - \beta_x) \times A_d^{ideal}(x) + \beta_x \times A_{1+1}(x) \tag{5.7}$$

Maintenance time on link x failure can be calculated as follows. First we compute the blocking probability β_x by computing dynamic paths for flows in failed link x. Then, we define constant MTBF value and give variable MTTR value m. At this point, the availability function when link x fails is denoted as $A_d(x, m)$ with parameters x and m. Maintenance time for achieving availability $A_d(x, m)$ is corresponds to m. Therefore, when our objective availability (e.g., 99.9999%) is given as A_{obj}, maximum value m whose $A_d(x, m)$ does not fall below A_{obj} becomes maintenance time, and it is given as follows;

$$MT(x) = max_{A_{obj} \leq A_d(x,m)}(m) \tag{5.8}$$

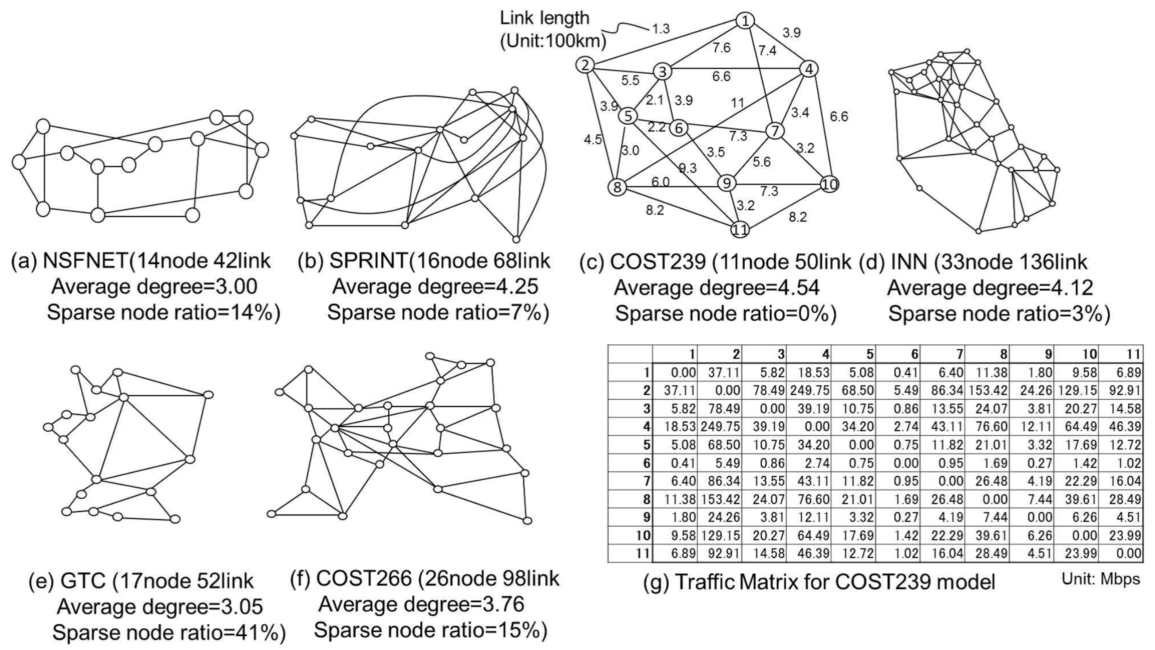
5.5 Performance Evaluation

We show the effectiveness of our dynamic path reconfiguration through computer simulations. We evaluate our proposal considering two scenarios: (i) network design problem and (ii) re-design problem for operating network. The network design problem (i) outputs routes for each path and required bandwidth for each link when the physical topology and traffic matrix are given. We evaluate our proposal compared to the three-path predesign approach, which constructs disjoint three paths using Suurballe algorithm [63]. The network re-design problem for operating network (ii), available bandwidth of each link was also given because upper bound of link bandwidth was defined, and primary and backup paths were established. In this problem, Suurballe algorithm cannot be applied because it computes N-disjoint path at once. Then, we evaluate our W-CSPF algorithm compared to benchmarks: SPF and CSPF described in Sec. 5.3.2.

This evaluation assumes possible single link failure occurrence. The blocking probability of new backup paths after a link x fails is given as follows.

$$\beta_x = 1 - \frac{|F'_x|}{|F_x|} \tag{5.9}$$

$|F_x|$ is a number of paths included on link x, and $|F'_x|$ is a number of paths, which can provide the third path. The third path means the third backup path for the



	1	2	3	4	5	6	7	8	9	10	11
1	0.00	37.11	5.82	18.53	5.08	0.41	6.40	11.38	1.80	9.58	6.89
2	37.11	0.00	78.49	249.75	68.50	5.49	86.34	153.42	24.26	129.15	92.91
3	5.82	78.49	0.00	39.19	10.75	0.86	13.55	24.07	3.81	20.27	14.58
4	18.53	249.75	39.19	0.00	34.20	2.74	43.11	76.60	12.11	64.49	46.39
5	5.08	68.50	10.75	34.20	0.00	0.75	11.82	21.01	3.32	17.69	12.72
6	0.41	5.49	0.86	2.74	0.75	0.00	0.95	1.69	0.27	1.42	1.02
7	6.40	86.34	13.55	43.11	11.82	0.95	0.00	26.48	4.19	22.29	16.04
8	11.38	153.42	24.07	76.60	21.01	1.69	26.48	0.00	7.44	39.61	28.49
9	1.80	24.26	3.81	12.11	3.32	0.27	4.19	7.44	0.00	6.26	4.51
10	9.58	129.15	20.27	64.49	17.69	1.42	22.29	39.61	6.26	0.00	23.99
11	6.89	92.91	14.58	46.39	12.72	1.02	16.04	28.49	4.51	23.99	0.00

(g) Traffic Matrix for COST239 model Unit: Mbps

Figure 5.3: Simulation topologies and traffic matrix for COST239 model.

existing three-path predesign, and the dynamic path for our dynamic reconfiguration. For the parameters for computing maintenance time, MTBF is set to three years, and A_{obj} is set to 99.9999% availability.

Evaluation topologies and traffic matrix are illustrated in Fig. 5.3. Distribution of traffic matrix is according to gravity model [74] whose distribution is proportional to weight of nodes. We assume that blocking probability of path is severely dependent on the distribution of node-degree. Then, evaluation is performed on six different topologies whose average node-degrees are over three. For both the dynamic path configuration and pre-design method, another path (dynamic path or third path) is additionally established to the current 1+1 path protection. Therefore, the ratio of nodes whose node-degree is under three becomes main factor of performance. Then, we define the node whose node-degree is under three as *sparse node*. For the evaluation of equipment cost and path length for the network design problem (i), we evaluate them only on (c) COST239 model, whose minimum cut is over three, to ignore the effect of blocking probability. In addition, for the re-design problem (ii), we evaluate blocking probability on (c) COST239 model for clarifying the difference in performance caused by the different path computation algorithms. Our proposal is assumed to be applied to the electric MPLS network [3]. Each path can request the bandwidth with an arbitrary granularity. In each physical link, only an upper bound of link capacity is given: we do not deal the wavelength constraints [75] for optical network.

Table 5.3: Total amount of required resources and path lengths on cost239 model.

	Total Bandwidth			Average Path Length		
	1st	2nd	3rd/Dynamic	1st	2nd	3rd/Dynamic
Dedicated Protection	1	1.235	1.802	1	1.349	1.872
Shared Protection	1	1.235	1.383	1	1.349	1.872
Proposed Protection	0.9980	1.233	1.439	0.9964	1.343	1.730

5.5.1 Effectiveness under the Network Design Problem

In this section, we evaluate total equipment cost, path length, and dependency to the form of network topology for the network design problem.

Equipment Cost and Path Length

Table 5.3 shows the total bandwidth and path length required for primary paths, backup paths, and third paths for the existing three-path predesign and proposed dynamic protection respectively. The total bandwidth means the summation of bandwidth of each link. The total bandwidth and path length are normalized so that those of the dedicated protection method become one. There are two types of the three-path predesign method in terms of bandwidth design: the dedicated protection whose third paths occupy their required bandwidth, and the shared protection whose third paths, which have different source and destination node pair, share the bandwidth each other.

From the table 5.3, we can find the trade-off between the total bandwidth and the path length: the total bandwidth of our dynamic protection increases about 4.0% compared to the shared protection, and the path length is shortened about 3.6%. The reason of the total bandwidth increase is because of *path spreading*: because our dynamic protection behaves as the link protection mechanism, the dynamic paths are provided as different routes in accordance with the failed point even if those paths have same source and destination node. On the other hand, the three-path predesign method provides the third path, which does not share links or nodes with primary and backup path, between the same source and destination node. That is, with three-path predesign, the third path for the same source and destination does not change in accordance with the failed point. As a result, the required bandwidth of our dynamic protection slightly increases compared to the three-path predesign.

Concerning the path length, our dynamic protection can shorten the length of primary path, backup path, and third path compared to them of three-path predesign method. In the three-path predesign method, a primary path, a backup path, and a third path are computed as disjoint paths each other. With our dynamic protection, only a primary path and a backup path are computed as disjoint paths. A dynamic path is computed as disjoint path against the new primary path, and failed link or node. As a result, the restriction for establishing the primary path, the backup path, and the dynamic path is relaxed, and this

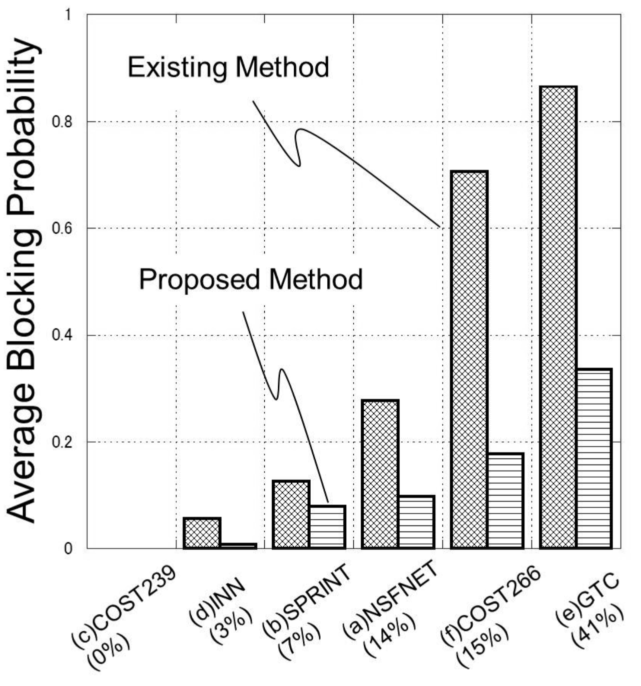


Figure 5.4: Average blocking probability of the dynamic path reconfiguration and three-path predesign design method. The number in brackets means the sparse node ratio.

result in shortening each path compared to the three-path predesign. This effect can also reduce the required bandwidth in the partial protection environment, which only protects a part of links or nodes.

Blocking Probability

Figure 5.4 illustrates the blocking probability of our dynamic path reconfiguration and three-path predesign under the different topologies. In this evaluation, we measure the average blocking probability when a possible single link failure occurs.

While the blocking probability with the three-path predesign increases as the ratio of sparse nodes increases, our dynamic protection suppresses the increase of blocking probability. For example, on (f) COST266 model and (e) GTC model, blocking probability with ours is reduced about 53% compared to the three-path predesign. This is because that our dynamic protection maximally avoids the influence of sparse node existence by dynamically discovering the third path. The three-path predesign method could not provide a third path if the primary or backup path between a source node and destination node include at least one sparse node: Even if a failure occurs on a link which does not connect the sparse node, the third path could not be provided. On the other hand, our dynamic protection, which behaves like a link protection, could provide the third path excepting the case where the link connected with the sparse node fails.

The blocking probability also depends on the number of nodes, and suppressing effect increases as the number of nodes increases. For example, on (a) 14-nodes

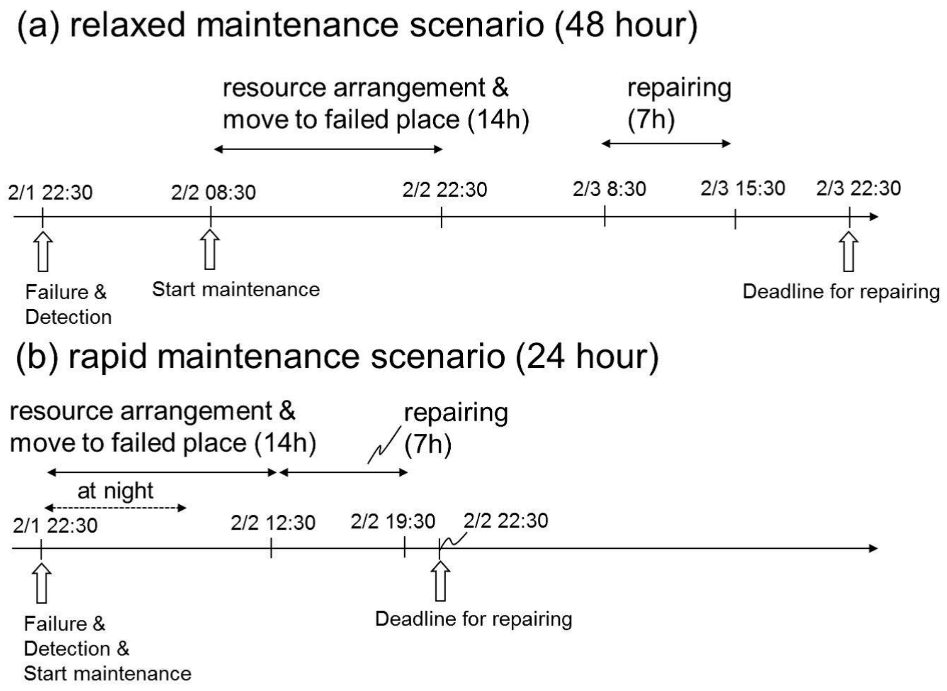

(a) relaxed maintenance scenario (48 hour)
resource arrangement & move to failed place (14h)
repairing (7h)
2/1 22:30
2/2 08:30
2/2 22:30
2/3 8:30
2/3 15:30
2/3 22:30
Failure & Detection
Start maintenance
Deadline for repairing
(b) rapid maintenance scenario (24 hour)
resource arrangement & move to failed place (14h)
repairing (7h)
at night
2/1 22:30
2/2 12:30
2/2 19:30
2/2 22:30
Failure & Detection & Start maintenance
Deadline for repairing

Figure 5.5: Simulation Scenario for failure repair. The maintenance time of (a), and (b) is over 48 hour and 24 hour respectively.

NSFNET model and (f) 26-nodes COST266 model, while both the ratios of sparse nodes are in the same range (14-15%), suppression effect of (f) COST266 is larger than it of (a) NSFNET. The suppression effect of (f) COST 266 is about 53%, and is equivalent to it of (e) GTC model, whose ratio of sparse node is about 41%. Because the total numbers of links and nodes of (f) COST266 model is greater than them of (a) NSFNET, the number of intermediate nodes of (f) COST266 is also greater than them of (a) NSFNET. That is, with three-path predesign method, the number of failure patterns, which could not provide third path, increases even if the ratio of sparse node is almost same. On the other hand, blocking probability with our dynamic protection is affected by only the ratio of sparse node because of its link protection behavior. As a result, in our dynamic protection, increase of intermediate nodes has no relevance to the increase of blocking probability, and then suppression effect, which also means the relative effect to the three-path predesign, increases as the number of nodes increases.

5.5.2 Effectiveness in Operating Network

In this section, we evaluate the effectiveness of our W-CSPF algorithm under the re-design scenario for operating networks compared to benchmarks: CSPF and SPF. Evaluation indexes are blocking probability and maintenance time. For discussing the right and wrong of maintenance time, we consider specific scenarios shown in Fig. 5.5. Figure 5.5 (a) illustrates a failure repair scenario that has over 48 hours for maintenance, and Fig. 5.5 (b) illustrates it that only has 24 hours.

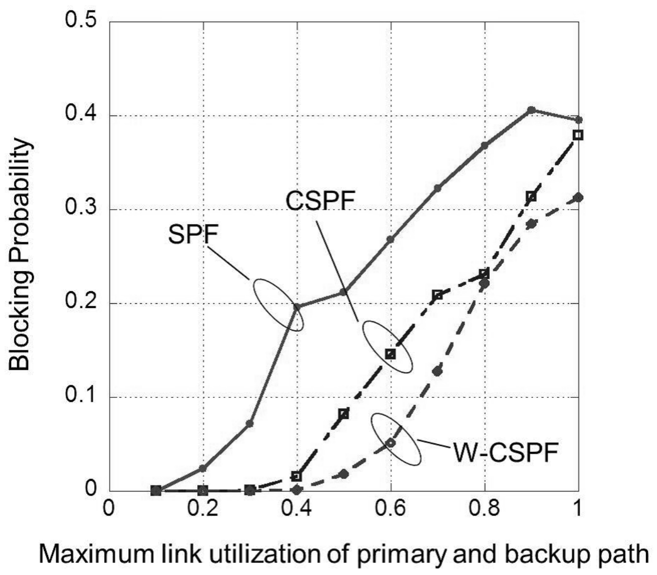


Figure 5.6: Blocking probability with different dynamic path protection algorithms on cost239 model.

In these scenarios, we assume that it takes 14 hours to arrange new equipment and to go off to failed place, and 7 hours to repair the failure. In addition, we assume that a failure occurs and is notified to network operator after office hours (22:30 P.M.). If allowed maintenance time is 48 hours (Fig. 5.5 (a)), we can repair the failure without working at late-evening. On the other hand, if the allowed maintenance time is 24 hours (Fig. 5.5 (b)), we should rapidly go to the failed equipment with late-evening work. We therefore consider allowing 48-hour repair time is preferable over 24-hour one.

Blocking Probability with different algorithms

Figure 5.6 illustrates blocking probability with different dynamic path protection algorithms. Blocking probability is computed as an average when a possible single link failure occurs. In this evaluation, we changed maximum link utilization as variable to evaluate relationship between dynamic path computation algorithms and the available resources. First, we accommodate traffic demands (Fig. 5.3) on primary and backup paths, which are computed by Suurballe algorithm. Then, link bandwidth of each link is adjusted to satisfy the given maximum link utilization.

The reduction effect of blocking probability with our W-CSPF compared to CSPF and SPF becomes high when network is moderately-loaded: the reduction effect is about 20% compared to SPF, and is about 10% compared to CSPF when maximum link utilization is between 0.5 and 0.7. When the link utilization is low, there are enough available resources. Then, effect of taking available resources into consideration is low. For example, SPF algorithm also realizes zero blocking when maximum link utilization is 0.1. When maximum link utilization is between 0.2 and 0.3, while the path blocking arises with SPF, CSPF algorithm, which only eliminates unsatisfied links, can still realize zero blocking. When maximum link

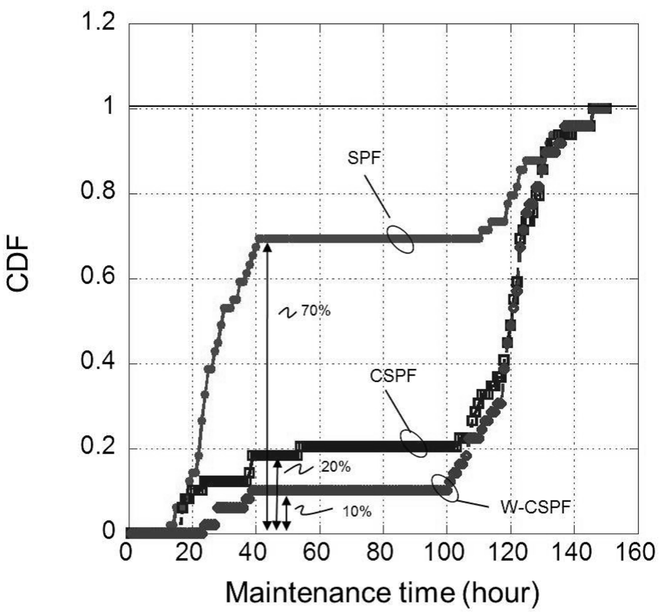


Figure 5.7: Figure shows the cumulative distribution function (CDF) of maintenance time when maximum link utilization is 0.5 on COST239 topology model.

utilization is between 0.4 and 0.7, on the other hand, available resources gradually decrease, and then performance difference between CSPF and W-CSPF appears. This is caused by the consideration of available resources: while CSPF algorithm only eliminates the unsatisfied links, W-CSPF not only eliminates the unsatisfied links, but also utilizes available links by weight assignment. Finally, when the maximum link utilization is between 0.8 and 1.0, there are few available resources in the network. Most of links have no available resources, and then available links for dynamic path accumulation is restricted. As a result, dynamic paths computed by CSPF and W-CSPF algorithms are stochastically the same routes. Then, performance difference between CSPF and W-CSPF is small.

Because the main factor for determining the performance of each algorithm is the amount of available resources, dependency to traffic distribution is assumed to be small. Even if traffic distribution affects the amount of available resources, performance difference, which has roots in the mechanism of available resources consideration, does not change. Then, the trend of the results will not change even though we evaluate performance under different traffic distribution models.

Relaxation of Maintenance Time

The cumulative distribution function (CDF) of maintenance time based on the blocking probability of each algorithm when maximum link utilization is 0.5 is illustrated in Fig. 5.7. This graph suggests that the path computation algorithm with low blocking probability also decreases the ratio of failures that have short maintenance time. For example, while the ratio of failures whose maintenance time is under 48 hours is about 70% with SPF based algorithm, it becomes about

20% with CSPF based one, and it becomes 10% with W-CSPF based one. The cumulative value of each algorithm increases when maintenance time is around 20-40 hours, and 100-140 hours. The former increasing is for the case that the dynamic path cannot be established, and the latter is for the case that the dynamic path is established. With W-CSPF algorithm, the cumulative value increases around 100 hours. This also suggests that maintenance time over 100 hours is assigned to 90% of failures. The tendency of CSPF and W-CSPF is resemblant, and cumulative value of W-CSPF temporarily exceeds it of CSPF around about 120 hours. This is because that the length of dynamic path with W-CSPF, that considers efficient utilization but not considers path length, becomes slightly long compared to the CSPF at this point. However, this temporal exceeding is not so important compared to the advantage of W-CSPF: it can decrease the ratio of failures with short maintenance time about 10% compared to the CSPF based.

5.6 Conclusion

In this chapter, we proposed the network architecture and path computation algorithm to maintain 1+1 path protection even after a single failure by dynamically assigning a new backup path. In terms of path quality, there was the trade-off between the equipment cost and the path length: the equipment cost increased about 4.0%, and the path length was shortened about 3.6% compared to the existing static design. Our dynamic path protection, which maximally avoided the influence of sparse-node existence, could reduce blocking probability about 53% compared to the static design. In the re-design scenario for operating networks, our algorithm called W-CSPF could reduce blocking probability about 10%-20% compared to simple benchmark algorithms. In addition, we formulated the relation between maintenance time and blocking probability. We showed the effectiveness in terms of relaxation of maintenance time: ratio of failures, whose maintenance time could be assigned over 100 hours, increased about 10%.

Chapter 6

Conclusions

This study provided reliable and sustainable network design for IP and MPLS network. In particular, we studied carrier grade IP fast rerouting and a scheme to relax repair-time constraint for MPLS network.

In Chapter 2, we argued that minimizing the number of backup topologies for IP fast rerouting is required to improve network scalability and applicability. We presented a new backup topology computation algorithm that reduces the number of backup topologies. Our algorithm has high feasibility for the existing IP fast rerouting framework in terms of the memory consumption, and we demonstrated its effectiveness through an extensive simulation study. The results showed that our algorithm reduced the number of backup topologies by about 56% on power-law networks, and the effectiveness increases as the number of nodes increases. Our algorithm is more effective on actual networks than the conventional algorithm.

In Chapter 3, we tackled the minimization of the loop probability of the IP fast rerouting on concurrent double failures. Our forwarding algorithm, which does not require explicit failure notification, can reduce the loop probability from 10^{-2} to 10^{-3} compared to existing algorithms. To achieve the objective loop probability on an arbitrary size network, our algorithm can reduce the required number of backup topologies about 35-50% compared to the loose lower bound solutions, where arbitrary two-link pairs are protected on one of the backup topologies. Because our algorithm can avoid explicit failure notification, and the increase in the number of backup topologies as network size grows is slower than existing algorithms, it has high scalability.

In Chapter 4, we proposed the autonomous IP fast rerouting method using OpenFlow. By utilizing OpenFlow, we can achieve sub-50ms restoration without any extension of current forwarding hardware. In addition, our flow table compression algorithm can reduce the size of flow table about 45% compared to existing algorithms. This compression effect also increases the feasibility of IP fast rerouting on actual networks.

In Chapter 5, we proposed the network architecture and path computation algorithm to maintain 1+1 path protection as much as possible after a single failure by dynamically assigning a new backup path. In terms of path quality, there was the trade-off between the equipment cost and the path length: the equipment cost increased about 4.0%, and the path length was shortened about 3.6% compared to the existing static design. Our dynamic path protection, which maximally avoided the influence of sparse-node existence, could reduce blocking probability about 53% compared to the static design. In the re-design scenario for operating networks, our algorithm called W-CSPF could reduce blocking probability about 10%-20% compared to simple benchmark algorithms. In addition, we formulated the relation between maintenance time and blocking probability. We showed the effectiveness in terms of relaxation of maintenance time: ratio of failures, whose maintenance time could be assigned over 100 hours, increased about 10%.

The conclusion of this dissertation is that IP and MPLS are both essential technologies in accordance with network operation policy and/or service quality requirements, and this study expands the applicability of each technology. Our IP fast rerouting not only satisfies carrier requirements but also provides cost-effective restoration, which is originally derived from IP nature. In addition, our MPLS protection framework is extended for realizing relaxed maintenance requirement while its potential for traffic engineering is also assured.

Finally, we conclude this dissertation by mentioning future works. For the algorithms of backup topology design in Chap. 2 and Chap. 3, we will mathematically analyze them: proof the problem as NP hard and find suboptimal minimum number of backup topologies. For the IP fast rerouting framework in Chap. 4, we will implement our IP fast rerouting method for proof of concept, and develop the recovery optimization framework that uses both local and global repairing as the situation demands. In Chap. 5, we present one of applications for applying dynamic nature to path protection scheme. We believe that this new concept will motivate a rich body of research for the current static carrier network against more complicated failures such as multiple failures or disaster.

Bibliography

[1] T. Murakami, "The NGN - a carrier-grade IP convergence network," in Proc. of IEEE/IFIP NOMS Workshops, 2010.

[2] J. Moy, "OSPF Version 2," IETF RFC 2328, Apr. 1998.

[3] V. Sharma and F. Hellstrand, "Framework for Multi-protocol Label Switching (MPLS)-based Recovery," in IETF, RFC 3469, Feb. 2003.

[4] M. Bocci, et al., "A Framework for MPLS in Transport Networks," in IETF RFC 5921, Jul. 2010.

[5] S. Ramamurthy and B. Mukherjee, "Survivable WDM Mesh Networks, Part I - Protection," in Proc. of IEEE INFOCOM, vol. 2, pp. 744-751, Mar. 1999.

[6] C. Alaettinoglu, V. Jacobson, and H. Yu, "Towards Millisecond IGP Convergence," in IETF Internet draft 2000.

[7] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni, "Incremental Algorithms for Single-Source Shortest Path Trees," in Proc. of Foundations of Software Tech. and Theoretical Comp. Sci., Dec. 1994, pp. 113-24.

[8] Bernard Fortz, Mikkel Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," in Proc. of IEEE INFOCOM 2000.

[9] Yufei Wang, Zheng Wang, Leah Zhang, "Internet Traffic Engineering without Full Mesh Overlaying," in Proc. of IEEE INFOCOM 2001.

[10] A. Nucci et al., "IGP Link Weight Assignment for Transient Link Failures," in Proc. of Elsevier ITC 18, 2003.

[11] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," in Proc. of IEEE JSAC, vol. 20, no. 4, May 2002, pp. 756-67.

[12] P. Narvaez, "Routing Reconfiguration in IP Networks," Ph.D. dissertation, MIT, June 2000.

[13] Y. Rekhter et al., "A Border Gateway Protocol 4 (BGP-4) ," in IETF RFC4271.

[14] A. Bremler-Barr, Y. Afek, and S. Schwarz, "Improved BGP convergence via ghost flushing," in Proc. of IEEE INFOCOM, March 2003.

[15] J. Chandrashekar, Z. Duan, Z. Zhang, and J. Krasky, “Limiting Path Exploration in BGP,” in Proc. of IEEE INFOCOM 2005, vol.4, pp.2337-2348, March 2005.

[16] P. Marques, et al., “Advertisement of the best-external route to IBGP,” in IETF draft-marques-idr-best-external-00.txt, 2008.

[17] P. Mohapatra , et al., “Fast Connectivity Restoration Using BGP Add-path,” in IETF draft-pmohapat-idr-fast-conn-restore-00.txt 2008.9

[18] D.Walton, et al., “Advertisement of Multiple Paths in BGP,” in IETF draft-walton-bgp-add-paths-06, 2009.

[19] C. Pelsser et al., “Improving Route Diversity through the Design of iBGP Topologies,” in Proc. of IEEE ICC 2008.

[20] N. Kushman, S. Kandula, D. Katabi, and B. M. Maggs, “R-BGP:Staying connected in a connected world,” in Proc. of NSDI’07 April 2007.

[21] W. Xu and J. Rexford, “Miro: multi-path interdomain routing, ” in Proc. of ACM SIGCOMM 2006, September 2006.

[22] Masafumi Watari, Yuichiro Hei, Shigehiro Ano,Katsuyuki Yamazaki, “OSPF-based Fast Reroute for BGP Link Failures,” in Proc. of IEEE globecom 2009.

[23] Ning Wang, Yu Guo, Kin-Hon Ho, Michael Howarth,George Pavlou, “Fast Network Failure Recovery Using Multiple BGP Routing Planes,” in Proc. of IEEE globecom 2009.

[24] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, “Fast IP Network Recovery using Multiple Routing Configurations,” in Proc. of INFOCOM, Apr. 2006.

[25] T. Cicic, A. F. Hansen, A. Kvalbein, M. Hartmann, R. Martin, and M. Menth, “Relaxed Multiple Routing Configurations for IP Fast Reroute,” in Proc. of IEEE/IFIP Network Operations and Management Symposium 2008.

[26] S. Kamamura, T. Miyamura, C. Pelsser, I. Inoue, and K. Shiomoto, “Minimum Backup Configurations Creation Method for IP Fast Reroute,” in Proc. of IEEE Globecom, Dec. 2009.

[27] A. F. Hansen, O. Lysne, T. Cicic, and S. Gjessing, “Fast Proactive recovery from Concurrent Failures,” in Proc. of IEEE International Conference on Communications, ICC 2007, June 2007.

[28] A. Kvalbein, T. Cicic and S. Gjessing, “Post-Failure Routing Performance with Multiple Routing Configurations,” in Proc. of INFOCOM, May 2007.

[29] R. Takahashi, S. Tembo, K. Yukimatsu, S. Kamamura, T. Miyamura, and K. Shiomoto, “Dispersing Hotspot Traffic in Backup Topology for IP Fast Reroute,” in Proc. of ICC, Jun. 2011.

[30] A. Atlas and A. Zinin, "Basic specification for IP fast reroute: Loop-free alternates," in IETF RFC 5286, Sep. 2008.

[31] S. Bryant, et al., "IP Fast Reroute Using Not-via Addresses," in IETF draft-ietf-rtgwg-ipfrr-notvia-addresses-08, Dec. 2011.

[32] S. Nelakuditi et al., "Failure Insensitive Routing for Ensuring Service Availability," in Proc. of IW QoS, June 2003.

[33] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C.-N. Chuah, "Failure inferencing based fast rerouting for handling transient link and node failures," in Proc. of IEEE Global Internet, vol. 4, Mar. 2005.

[34] J. Wang, and S. Nelakuditi, "IP Fast Reroute with Failure Inferencing," in Proc. of INM'07, at ACM SIGCOMM, Aug. 2007.

[35] Kang Xi, H. J. C. "ESCAP: Efficient SCan for Alternate Paths to Achieve IP Fast Rerouting" in Proc. of IEEE Globecom 2007.

[36] A. Tam, K. Xi, and H. J. Chao, "A Fast Reroute Scheme for IP Multicast," in Proc. of IEEE Globecom, Dec. 2009.

[37] Kang Xi, C. H. J., C. Guo, "Recovery from Shared Risk Link Group Failures Using IP Fast Reroute," in Proc. of IEEE ICCCN Aug. 2010.

[38] M. Gjoka, V. Ram, and Y. Xiaowei, "Evaluation of IP Fast Reroute Proposals," in Proc. of COMSWARE Jan. 2007.

[39] P. Psenak, S. Mirtorabi, A. Roy, L. Nguen, and P. Pillay-Esnault, "MT-OSPF: Multi topology (MT) routing in OSPF," in IETF RFC4915, June 2007.

[40] H. Liu, "Routing Table Compaction in Ternary CAM," in IEEE Micro, Vol.22,No.1, pp.58-64, Jan./Feb. 2002.

[41] K. Nichols, et al., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," in IETF RFC 2474, Dec. 1998.

[42] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," in Proc. of the American Mathematical Society, Vol. 7, No. 1, Feb. 1956, pp. 48–50.

[43] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An approach to universal topology generation," in Proc. of IEEE MASCOTS, Aug. 2001.

[44] J.T. Moy, OSPF: Anatomy of an Internet Routing Protocol, Addison Wesley, 1998.

[45] C. Berge, The Theory of Graphs, New York: Dover Publications, 2001.

[46] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An Information Theoretic Approach to Traffic Matrix Estimation," in Proc. of IEEE SIGCOMM 2003.

[47] S. Kamamura, T. Miyamura, and K. Shiomoto, "IP Fast Reroute Control using Centralized Control Plane Architecture," in Proc. of IEEE CNSM, Oct. 2010.

[48] F. K. R. Chung, "Spectral Graph Theory," volume 92 of Regional Conference Series in Mathematics. American Mathematical Society, Providence, RI, 1997.

[49] B. Fortz, "Application of Meta-heuristics to Traffic Engineering in IP Networks," International Transactions in Operational Research, Vol. 18, Issue 2, pp.131-147, Mar. 2011.

[50] "Multi-Topology routing," Web site, http://www.juniper.net/us/en/local/pdf/whitepapers/2000308-en.pdf, access to May 2012.

[51] "Open VSwitch," Web site, http://openvswitch.org/, access to May 2012.

[52] "GNU Quagga Project, " Web site, http://www.quagga.org/, access to May. 2012.

[53] A. Greenberg, et al., "A Clean Slate 4D Approach to Network Control and Management," in Proc. of ACM SIGCOMM Computer Communication Review. 35(5). Oct. 2005.

[54] N. McKeown, et al., "OpenFlow: enabling innovation in campus networks," in Proc. of ACM SIGCOMM Computer Communication, 38(2):69-74, April 2008.

[55] "OpenFlow Switch Specification 1.1," Web site, http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf, access to May 2012.

[56] J.-P. Vasseur, M. Pickavet, and P. Demeester. "Network Recovery: Protection and Restoration of Optical, SONET-SDH, and MPLS" Morgan Kaufmann, 2004.

[57] P. Francois et al., "Achieving sub-second IGP convergence in large Ipnetworks," in Proc. of ACM SIGCOMM Computer Communication Review Volume 35 Issue 3, July 2005

[58] D. Oran. "OSI IS-IS intra-domain routing protocol," in IETF RFC 1142, Feb. 1990.

[59] "Example: Configuring the OSPF Routing Algorithm," Web site, http://www.juniper.net/techpubs/en_US/junos12.1/topics/topic-map/ospf-spf-algorithm.html, access to May 2012.

[60] C. Filsfils, P. Mohapatra, J. Bettink, P. Dharwadkar, P. De Vriendt, Y. Tsier, V. V. D. Schrieck, O. Bonaventure and P. Francois, "BGP Prefix Independent Convergence (PIC) Technical Report," Tech. Rep. 2011. Web site, http://www.cisco.com/en/US/prod/collateral/routers/ps5763/bgp_pic_technical_report.pdf, access to May 2012.

[61] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, C. Diot, “Characterization of Failures in an IP Backbone,” in Proc. of IEEE INFOCOM, vol. 4, pp. 2307-2317.Mar. 2004.

[62] P. H. Ho and H. T.Mouftah, “Shared protection in WDM mesh networks,” in IEEE Commun. Mag., vol. 42, no. 1, pp. 70-76, Jan. 2004.

[63] J. W. Suurballe, “Disjoint paths in a network,” Networks, vol. 4, ppp. 125-145, 1974.

[64] P. Soproni, P. Babarczi, J. Tapolcai, T. Cinkler, and P. H. Ho, “A Meta-Heuristic Approach for Non-Bifurcated Dedicated Protection in WDM Optical Networks,” in Proc. of Design of Reliable Communication Networks, (DRCN), 2011, pp. 110-117.

[65] P. Babarczi, J. Tapolcai, and P. Ho, “Availability-constrained Dedicated Segment Protection in circuit switched mesh networks,” in Workshop on Reliable Networks Design and Modeling, (RNDM), 2009, pp. 1-6.

[66] T. Cinkler and L.Gyarmati, “MPP optimal multi-path routing with protection,” in Proc. of IEEE ICC 2008, pp. 165-169.

[67] D. Katz, K. Kompella, and D. Yeung, “Traffic engineering (TE) extensions to OSPF version 2,” in IETF RFC 3630, Sept. 2003.

[68] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, “RSVP TE : extensions to RSVP for LSP tunnels,” in IETF RFC 3209, Dec. 2001.

[69] J. Vasseur, and J. L. Roux, “Path Computation Element (PCE) Communication Protocol (PCEP),” in IETF RFC 5440, Mar. 2009.

[70] A. Farrel, et al., “A Path Computation Element (PCE)-Based Architecture,” in IETF RFC 4655, Aug. 2006.

[71] L. Yang et al, “Forwarding and Control Element Separation (ForCES) Framework,” in IETF RFC 3746 Apr. 2004.

[72] “OpenFlow Discovery Protocol and Link Layer Discovery Protocol,” http://groups.geni.net/geni/wiki/OpenFlowDiscoveryProtocol, access to Apr. 2012.

[73] M, KB, “Constrained Shortest Path First,” in IETF internet draft, draft-manayya-constrained-shortest-path-first-02.txt, Feb. 2010.

[74] Y. Zhang, M. Roughan, N. Duffield and A. Greenberg, “Fast accurate computation of large-scale IP traffic matrices from link loads,” in Proc. of ACM SIGMETRIC’03, pp.206-217, June 2003.

[75] L. Sahasrabuddhe, S. Ramamurthy, and B. Mukherjee, “Fault Management in IP-Over-WDM Networks: WDM Protection Versus IP Restoration, ” in IEEE JSAC, Vol. 20, No. 1, Jan. 2002.

Research Achievements

Journal Papers

1. Autonomous IP Fast Rerouting with Compressed Backup Flow Entries using OpenFlow, IEICE Trans. INF. & SYST., Vol.E96-D, No.2, pp. 184-192, February 2013, Shohei Kamamura, Daisaku Shimazaki, Atsushi Hiramatsu, and Hidenori Nakazato
2. 動的なパス設定制御に基づく 1+1 パスプロテクション維持方式の提案，電子情報通信学会論文誌 (B)，Vol.J96-B, No.2, pp.48-58, February 2013, 鎌村 星平，島崎 大作，平松 淳，中里 秀則
3. Loop-free Fast Rerouting considering Double-link Failures, IEICE Trans. Commun. Vol.E95-B, No.12, pp.3811-3821, December 2012, Shohei Kamamura, Daisaku Shimazaki, Atsushi Hiramatsu, and Hidenori Nakazato
4. Scalable Backup Configrations Creation for IP Fast Reroute, IEICE Trans. Commun. Vol.E94-B No.1 pp.109-117, January 2011, Shohei Kamamura, Takashi Miyamura, Yoshihiko Uematsu, and Kohei Shiomoto

International Conference Papers

1. Minimizing loop occurrences for IP Fast Rerouting considering Double-link Failures, submitted to International Conference on Communications (ICC) 2013, Shohei Kamamura, Daisaku Shimazaki, Koji Sasayama, and Hidenori Nakazato.
2. Relaxed Maintenance Protection Architecture by Dynamic Backup Path Configuration, in Proc. of Optical Fiber Communication Conference (OFC) 2011, March 2011, Shohei Kamamura, Takashi Miyamura, and Kohei Shiomoto
3. Sticky 1+ 1 Path Protection Method by Dynamic Disjoint Path Discovery, in Proc. of Optical Networking Design and Modeling (ONDM) 2011, February 2011, Shohei Kamamura, Tomonori Takeda, Takashi Miyamura, Yoshihiko Uematsu, and Kohei Shiomoto
4. IP Fast Reroute Control using Centralized Control Plane Architecture, in Proc. of International Conference on Network and Service Management (CNSM) 2010, October 2010, Shohei Kamamura, Takashi Miyamura, and Kohei Shiomoto

5. Minimum Backup Configuration-Creation Method for IP Fast Reroute, in Proc. of Global Communications Conference (Globecom) 2009, December 2009, Shohei Kamamura, Takashi Miyamura, Cristel Pelsser, Ichiro Inoue, and Kohei Shiomoto
6. ScalableBackup Configurations Creation for IP Fast Reroute , in Proc. of IEEE Design of Reliable Communications Network (DRCN) 2009, October 2009, Shohei Kamamura, Takashi Miyamura, Cristel Pelsser, Ichiro Inoue and Kohei Shiomoto

Domestic and Japanese Conference Papers

1. [奨励講演] 二重故障環境下で IP Fast Rerouting を実現する予備トポロジー設計法の提案, 信学技報， Vol. 112, No. 85, pp. 37-42, 2012/6/22, 鎌村 星平，島崎 大作，平松 淳，中里 秀則
2. 多重故障を考慮したループフリー高速迂回方式に関する検討，信学技報， Vol. 111, No. 232, pp. 7-12, 2011/10/13, 鎌村 星平，島崎 大作，平松 淳，中里 秀則
3. [奨励講演]IP Fast Reroute 制御アーキテクチャの一検討, 信学技報, Vol. 110, No. 126, pp. 13-16, 2010/7/15, 鎌村 星平, 宮村 崇，植松 芳彦，塩本 公平
4. 動的パス制御による 1+1 パスプロテクション常時化方式の提案, 信学技報， Vol. 110, No. 93, pp. 1-6, 2010/6/24, 鎌村 星平、武田 知典、宮村 崇、植松 芳彦、塩本 公平
5. IP Fast Reroute のためのスケーラブルコンフィグレーション生成アルゴリズムの提案, 信学技報， Vol. 109, No. 129, pp. 67-72, 2009/7/17, 鎌村 星平、宮村 崇、ペルサー クリステル、井上 一郎、塩本 公平
6. OpenFlow Extension Framework for Autonomous Fast Recovery, IEICE General Conference Vol. 2012, BS-3-6,2012/3, Shohei Kamamura, Daisaku Shimazaki, Atsushi Hiramatsu, and Hidenori Nakazato
7. Minimizing Blocking Probability Problem under the Dynamic 1+1 Path Protection Environment, IEICE General Conference Vol. 2011, BS-4-39, 2011/3, Shohei Kamamura, Takashi Miyamura, and Kohei Shiomoto
8. Backup FIB Reduction Method using Shared Backup FIB for IP Fast Reroute, IEICE Society Conference Vol. 2010, BS-7-32, 2010/9, Shohei Kamamura, Takashi Miyamura, Yoshihiko Uematsu, and Kohei Shiomoto
9. Minimum Backup Configuration-Creation Method for IP Fast Reroute, IEICE Society Conference Vol. 2009, BS-10-3, 2009/9, Shohei Kamamura, Takashi Miyamura, Ichiro Inoue, and Kohei Shiomoto
10. A scalable backup configurations creation for IP-FRR, IEICE General Conference Vol. 2009, BS-4-5, 2009/3, Shohei Kamamura, Takashi Miyamura, Ichiro Inoue, and Kohei Shiomoto