Resilient Routing in the Internet

Suksant Sae Lor

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy
of the
University of London.

Department of Electronic & Electrical Engineering
University College London

I, Suksant Sae Lor, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

© 2006–2010, Suksant Sae Lor

Department of Electronic & Electrical Engineering

University College London

Abstract

Although it is widely known that the Internet is not prone to random failures, unplanned failures due to attacks can be very damaging. This prevents many organisations from deploying beneficial operations through the Internet. In general, the data is delivered from a source to a destination via a series of routers (*i.e* routing path). These routers employ routing protocols to compute best paths based on routing information they possess. However, when a failure occurs, the routers must re-construct their routing tables, which may take several seconds to complete. Evidently, most losses occur during this period.

IP Fast Re-Route (IPFRR), Multi-Topology (MT) routing, and overlays are examples of solutions proposed to handle network failures. These techniques alleviate the packet losses to different extents, yet none have provided optimal solutions. This thesis focuses on identifying the fundamental routing problem due to convergence process. It describes the mechanisms of each existing technique as well as its pros and cons. Furthermore, it presents new techniques for fast re-routing as follows.

Enhanced Loop-Free Alternates (E-LFAs) increase the repair coverage of the existing techniques, Loop-Free Alternates (LFAs). In addition, two techniques namely, Full Fast Failure Recovery (F3R) and fast re-route using Alternate Next Hop Counters (ANHC), offer full protection against any single link failures. Nevertheless, the former technique requires significantly higher computational overheads and incurs longer backup routes. Both techniques are proved to be complete and correct while ANHC neither requires any major modifications to the traditional routing paradigm nor incurs significant overheads. Furthermore, in the presence of failures, ANHC does not jeopardise other operable parts of the network.

As emerging applications require higher reliability, multiple failures scenarios cannot be ignored. Most existing fast re-route techniques are able to handle only single or dual failures cases. This thesis provides an insight on a novel approach known as Packet Re-cycling (PR), which is capable of handling any number of failures in an oriented network. That is, packets can be forwarded successfully as long as a path between a source and a destination is available. Since the Internet-based services and applications continue to advance, improving the network resilience will be a challenging research topic for the decades to come.

Acknowledgements

I would like to thank my grateful thanks to my *Role Model*, Dr Miguel Rio, for his invaluable guidance and inspiration throughout my studies. My special thanks go to Prof George Pavlou for his consistent hints and advice. My hearty thanks go to Dr David Griffin and Dr Jason Spencer for the great fun and experience under the AGAVE project. I owed Dr Richard Clegg immeasurable thanks for always saving me from the mathematical maze. My joyful thanks go to Dr Eleni Mykoniati for revitalising my soul by organising the mind-opening events and trips. I hereby, convey my heart-felt thanks to Dr Ning Wang for kindly providing me useful information and advice. I became brighter through great discussions and works with Dr Raul Landa and Mr Redouane Ali, you guys, please accept my thanks from the bottom of my heart. My sincere thanks go to Mr Lawrence Latiff for sparking great fun that tremendously alleviated tension during the conference trip in Japan.

I simple feel wordless in appreciation of the comfort, care, and sacrifice my family has given me from scratch to success, you are forever in my heart!

For my other colleagues and friends from NSRL, and those whose names are not mentioned, I thank you very so much for your kind words, encouragement, assistance, and advice in one way or another.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For decades, the Internet has become the largest data communication network connecting personal computers, servers, and many other Internet-enabled devices [20]. Statistically, it has more than 1.96 billion users as of June, 2010[1] and continues to grow. Nevertheless, services and applications operating online have different requirements. More precisely, some of them demand a high performance network, that can accommodate sufficient resources in order to provide certain Quality of Service (QoS). This is important, in particular for service providers who have signed contracts with customers, indicating that a certain level of service will be provided in exchange for the money paid. Thus, the demand for correct and fast data transfer through a high reliable network has been increasing continuously, to support emerging services that are not fault-tolerant.

This thesis focuses on how to achieve a resilient network using different routing strategies. It highlights the needs for a resilient network and the shortcomings of existing solutions with an extensive analysis. In general, routing can be categorised into intra-domain and inter-domain routing. The former signifies routing within a single logical area called an Autonomous System (AS) while the latter conveys routing across ASes. To find an optimal solution, the study of this thesis incorporates both aspects.

1.1 Objectives

Network reliability is one of the most important features in the present days. Emerging services involve a massive data transfer between end points. However, it is likely that traditional infrastructure originally designed by the ARPANET[2] will fail to accommodate these services due to its lack of resilient mechanisms. The main objectives of this thesis are *a)* to provide an extensive research on the Internet routing and its existing resilient mechanisms, and *b)* to provide alternatives as well as to find optimal solutions, that can elevate the network reliability without any major modifications to the traditional routing paradigm.

1.2 Motivation and Challenges

At the very beginning of the Internet era, one would not be concerned if packet forwarding is disrupted for a minute, especially if the connection between end points are oriented. If there is a topological

[1] `http://internetworldstats.com/stats.htm`.

[2] Advanced Research Projects Agency Network was one of the first packet switched networks.

change, the network will eventually converge and resume normal operations. Undelivered packets will be re-originated by their source again as no acknowledgements have been received. Classic applications such as e-mail, file transfer, and web surfing are well accommodated by the original Internet design. However, as new technologies and applications emerge, the Internet Service Providers (ISPs) and most Internet users start to doubt if they can always rely on the Internet for data communications. For example, incomplete (or very slow) data transfer caused by the forwarding discontinuation may concern the users of sensitive applications (*e.g.* remote surgery via the Internet).

Consequently, several approaches have been proposed to alleviate the routing disruption problem. However, as none of the techniques provide optimal solutions, the challenge to find the most suitable routing strategy still remains. Furthermore, resilient mechanisms must not only eliminate the forwarding disruption in the presence of failures or other unpredictable changes, but they must be scalable and have minimal impact on the operable parts of the network.

As the current networks employ a number of technologies under the same infrastructure, it is preferred that new techniques can be implemented without jeopardising the operability of other network functions. That is, a challenge to design a technique that requires no major modifications to the existing networks.

1.3 Contributions

The immediate beneficiaries of this research are the ISPs and the researchers in relevant areas. Extensive studies on routing and approaches for network resilience provide sufficient knowledge on the state of art of Internet routing. New techniques are also introduced in this thesis as alternatives for either practical implementations or experiments. Furthermore, these techniques can be tested in real networks without requiring any major modifications to the existing infrastructure.

In the long run, if these techniques are implemented, software developers and content providers will be the second tier beneficiaries. The deployment of resilient mechanisms will permit an additional range of sophisticated online applications that are not fault-tolerant. In addition, organisations involved in handling sensitive information such as hospitals and the army will be able to take full advantage of the network resilience. In the present day, it is difficult for hospitals to perform accurate remote health monitoring or surgery via the Internet because of unreliable networks. If the Internet transforms into a highly reliable and stable network, a huge range of sensitive applications will eventually have their place on the Internet.

1.4 Publications

- S. Sae Lor, R. Landa, and M. Rio. Packet Re-cycling: Eliminating Packet Losses due to Network Failures. In *Proc. ACM Workshop on Hot Topics in Networking (HotNets-IX)*, Monterey, CA, Oct 2010.

- S. Sae Lor and M. Rio. Enhancing Repair Coverage of Loop-Free Alternates. In *Proc. London Communications Symposium*, London, UK, Aug 2010.

- S. Sae Lor. A Novel Mechanism for Resilient Routing. In *Proc. Annual Workshop on Multi-Service Networks*, Abingdon, UK, Jul 2010.

- S. Sae Lor, R. Ali, R. Landa, and M. Rio. Recursive Loop-Free Alternates for Full Protection Against Transient Link Failures. In *Proc. IEEE Symposium on Computers and Communications*, Riccione, Italy, Jun 2010.

- S. Sae Lor, R. Landa, R. Ali, and M. Rio. Handling Transient Link Failures Using Alternate Next Hop Counters. In *Proc. IFIP Networking*, Chennai, India, May 2010.

- R. Ali, S. Sae Lor, and M. Rio. Two Algorithms for Network Size Estimation for Master/Slave Ad Hoc Networks. In *Proc. IEEE International Conference on Advanced Networks and Telecommunications Systems*, New Delhi, India, Dec 2009.

- R. Ali, S. Sae Lor, R. T. Benouer, and M. Rio. Cooperative Leader Election Algorithm for the Master/Slave Mobile Ad Hoc Network. In *Proc. IFIP Wireless Days*, Paris, France, Dec 2009.

- S. Sae Lor, R. Landa, and M. Rio. A New Technique for Full Fast Recovery in Hop-by-Hop Routing. In *Proc. London Communications Symposium*, London, UK, Aug 2009.

- S. Sae Lor. Achieving Resilient Routing in the Internet. In *Proc. Annual Workshop on Multi-Service Networks*, Abingdon, UK, Jul 2009.

1.5 Thesis Outline

The rest of this thesis is organised as follows. Chapter 2 provides an overview of the Internet routing that includes different algorithms used in existing routing protocols. It describes the routing within a single AS as well as routing between different ASes. Furthermore, brief details of both intra-domain and inter-domain routing protocols are also given. These include the Routing Information Protocol (RIP) [70] and Open Shortest Path First (OSPF) [90]. In addition, the Border Gateway Protocol (BGP) [103] is given as an example of widely deployed inter-domain routing protocol.

There is an evidence showing that the Internet has a certain level of resilience[3]. Nevertheless, it is not sufficient to handle intentional attacks, which can be potential threats in data communications. Therefore, different resilient approaches such as IP Fast Re-Route [113], Multi-Topology (MT) routing [63, 21], and overlay networks [4], used to eliminate the routing disruptions of a network are also presented.

Novel techniques namely, Enhanced Loop-Free Alternates (E-LFAs), Full Fast Failure Recovery (F3R), and fast re-route using Alternate Next Hop Counters (ANHC), are introduced in Chapter 3–5. Each technique has different benefits and trade-offs. A self-implemented Java software is used to simulate each routing technique on selective network topologies including real, inferred, and synthetic topologies. Furthermore, a number of evaluation metrics such as path characteristics and network load are used.

[3]The Internet is resilient to random failures [29, 23].

In Chapter 6, a new resilient routing approach using Packet Re-cycling (PR) mechanism is introduced. The technique can be employed in an oriented network and is capable of handling any number of failures as long as a path between a source and a destination exists. Chapter 7 concludes the thesis and discuss the future work.

Chapter 2

Resilient Routing

2.1 Introduction

The Internet is a very large packet switched network that has been growing continuously[1]. It interconnects an enormous number of networks allowing millions of devices to transfer data across the globe. Access to the Internet is not restricted only to business sectors or any particular industries, but is also available to residential and other public sectors. This creates a borderless data communications which is a pre-requisite for providing online services and contents. Nevertheless, as the access becomes more available, so does the demand for more advanced services and applications. A wide range of emerging services requires sophisticated platforms to accommodate different tasks and functionalities such as in e-commerce, e-health and some exhaustive resources consuming applications such as Video on Demand (VoD) and video conferencing. Although existing infrastructure can support classic services such as e-mail and basic file transfer very well, it fails to support sensitive applications such as remote surgery and military controls. Despite the demand's driver, whether it is the Service Level Agreement (SLA) or other factors, a proper distributed algorithm that is capable of retaining certain Quality of Service (QoS)[2] is required rather than relying on best effort delivery[3].

A process used to transfer data between any two points in the network is called *routing*. It involves the paths selection process to ensure correct delivery. However, a router may route packets based solely on the destination address depending on the protocol designs[4]. These designs specify the method for routers to utilise the information in the path selection process. Basically, the actual routing is performed based on routing tables built at routers running a routing protocol. However, the paths formed by these tables must be consistent throughout the network to guarantee a correct delivery. Each routing table consists of at least three information fields, which are: the network ID (destination ID), cost (path cost to the destination), and the next hop address (usually the address of an interface of a router) which indicates the next router of a path towards the destination. Nevertheless, a routing table may store additional information depending on the implementation of a routing protocol.

In practice, it means that a routing protocol used to construct routing tables has a great influence

[1] `http://internetworldstats.com/stats.htm`

[2] Ability to provide a certain level of performance to a data flow.

[3] No guarantee that data is delivered successfully.

[4] Traditional routing is based on destination address only.

upon the performance of a network. For example, using routing tables arrangement that allows faster best match operation will result in a better forwarding performance (*i.e.* faster delivery). In general, routing tables can be constructed as a Patricia tree[5] [87, 110], which minimises bit comparisons and requires only a single mask-and-match operation [26]. However, with certain designs, other algorithms such as balanced binary tree may be used. Furthermore, a hash table can be employed to speed up the routing table lookup process. Practically, a hash table[6] is employed by edge routers as a separate database used to search for particular destinations [91].

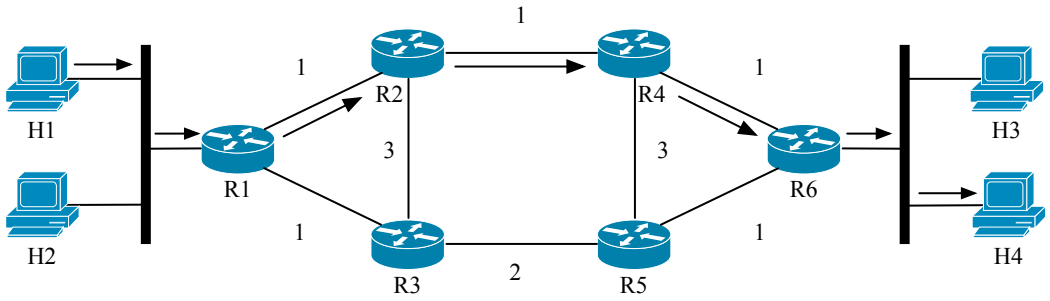


Figure 2.1: A simple routing example.

An example of a simple routing process is illustrated in Figure 2.1. Assume that all links connecting routers have the same characteristics (*i.e.* equivalent speed and capacity). Let H1 be the source device sending packets to H4. The host is connected to only one router, R1; hence, it becomes H1's first hop router[7]. The routing process starts when H1 forwards packets to R1. For each arriving packet, R1 determines the destination address stored in the packet header and performs a lookup process. Given that the routing table is already constructed, R1 forwards the packet to R2. This process iterates along the path R1→R2→R4→R6. Once the packet arrives at R6, it is forwarded to H4, which is the destination device.

As described earlier, the construction of routing tables may vary from one routing protocol to the others. For example, a router that employs a protocol based on the Shortest Path First (SPF) algorithm will normally construct its routing table that permits shortest paths forwarding for all source-destination node pairs. However, each routing protocol has different requirements and may be preferred differently based on network characteristics.

Due to practical concerns, routing protocols need to be designed to accommodate services and applications under different network environments. Some of which, may introduce more complexity and requirements. Since the Internet is composed of a large number of networks and the their equipment is manufactured by different vendors, the routing process introduces technical, operational, and management problems. When the network is operating under normal scenario, it is stable by design.

[5] A radix tree that contains large ranges of values with few exceptions suitable for IP addresses.
[6] Performs well only at the edge of the network, but not in the backbone areas due to overflows.
[7] The first router encountered on the path between the sending host and the destination host.

Nevertheless, this is not the case, as changes to the network topology due to failures, suspension, addition and removal of links or routers occur considerably frequently. These changes incur forwarding discontinuations and cause the traffic in transit to be dropped or caught in a loop. Thus, a routing protocol should be equipped with proper recovery mechanisms to return the network into a stable state.

In general, when a failure occurs, the router adjacent to the failure detects it and re-constructs the routing table. This involves the process of re-calculating the next hop for each prefix[8], called the *convergence process*. Furthermore, depending on the routing protocol, the failure-detecting router must notify all or a number of routers of the failure to ensure consistent forwarding paths. The time to complete the process is regarded as the *convergence time*.

As the reliability and stability of the network have direct impact on the network performance, fast convergence time is always preferred to avoid losses due to forwarding discontinuation. In practice, however, the convergence time may be significantly long for large and complex networks, which results in the performance being degraded [64].

The design goals of different routing protocols can be different depending on several factors. First, it has to be decided whether the administration of the protocol should be centralised or distributed. Although managing the routing information centrally may be more efficient in moderate size networks, it lacks of scalability. Second, a routing protocol may rely on the destination address only as in traditional routing paradigm, or employ the source address to enable certain mechanisms. For example, a strict source routing can be used to specify a path precisely. However, the packet overhead increases directly proportional to the network diameter. A loose source routing may be used to reduce this overhead as only a fraction of information on the path is embedded into each packet. Third, a routing protocol may either reacts to the network traffic deterministically or stochastically. Stochastic routing spreads the traffic among multiple paths (if applicable) to distribute the load [91].

Described above are fundamental concerns for a protocol design. However, as the demand for higher network reliability increases, disruption avoidance mechanisms also need to be considered. Several approaches have been proposed such as Multi-Topology (MT) routing [63, 21] which maintains different network topologies to provide different paths during failures. However, the technique requires larger routing table space and therefore, is not widely deployed in the Internet.

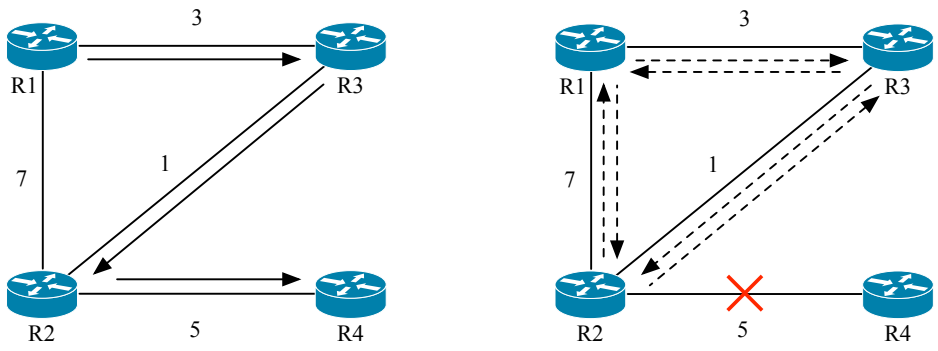
Classification of routing protocols can be divided into: *a)* static routing protocols and *b)* dynamic routing protocols. For a static routing protocol, routing tables need to be configured manually by network operators. Although it virtually requires no computational power, it becomes too difficult to manage in large networks. In contrast, dynamic routing is more adaptive to topological changes. For example, when a link fails, each router re-constructs its routing table automatically based on an updated network map. Despite this functionality, a skilled network operator is still required to perform proper configurations on protocol parameters. Evidently, common preferences for a routing protocol design are minimum overheads (*e.g.* complexity, memory, and control messages), maximum robustness, reliability, stability, and optimal paths [59]. Most of the times, it is unavoidable to lose certain features in exchange for some

[8] An address qualifier used to distinguish routes within an a network.

others.

2.1.1 Distance Vector Routing

Distance vector algorithms are asynchronous and distributed algorithms. The algorithms are employed by routers to compute the best paths to all destination prefixes. In general, these algorithms find the path that has a minimum cost metric. For example, Routing Information Protocol (RIP) [70] uses the number of hops as the metric. Thus, the path that has the least number of hops is considered as the best path. Nevertheless, the metrics employed by other distance vector routing protocols are not limited to the number of hops. The Interior Gateway Routing Protocol (IGRP)[9], for example, supports multiple metrics including bandwidth, delay, load, Maximum Transmission Unit (MTU) and the reliability of a path. Under the employment of a distance vector routing protocol, each router distributes its routing information, normally the current best paths, to its neighbours as well as receives the incoming advertisements. If any updates provide better paths than the local ones, a router updates its routing table and notifies its neighbours with the same information. This process continues until the network is stabilised (*i.e.* no update messages are sent or received). The main key feature of distance vector routing protocols is its simple implementation. Figure 2.2 illustrates an example of a distance vector routing.



(a) Routing from R1 to R4.

(b) Counting to infinity.

Figure 2.2: Routing based on distance vector algorithm.

An assumption that each router sends update messages to its neighbours synchronously is made for simplicity. Let T be the number of steps representing the process, the path computation based on a simple distance vector algorithm is described as follows:

- The algorithm initialises at $T = 0$. Each router creates a distance matrix indicating its immediate neighbours. As no other information available, the direct links connecting itself to the neighbour nodes form the best available paths.

- At $T = 1$, each router advertises its distance vector to all neighbours. Once the node receives the advertisements, a comparison is made with regard to the local information. If the updates provide better paths, a router updates its routing table and advertises its new distance vector to immediate

[9] A Cisco's proprietary routing protocol.

neighbours (except the one it receives the updates from). For example, R1 receives updates from R2 and R3, which provide better paths compared to the existing ones. Thus, R1 updates its routing table and advertises this information in the next step.

- The algorithm terminates at $T = 2$ where no more update messages are generated. Table 2.1 shows the steps of routing table construction at each router.

Table 2.1: Routing tables constructions based on distance vector algorithm.

Source	**R1**				**R2**			
Destination	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
$T = 0$	0, R1	7, R2	3, R1	∞	7, R1	0, R2	1, R2	5, R4
$T = 1$	0, R1	4, R3	3, R1	12, R2	4, R3	0, R2	1, R2	5, R4
$T = 2$	0, R1	4, R3	3, R1	9, R3	4, R3	0, R2	1, R2	5, R4
Source	**R3**				**R4**			
Destination	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
$T = 0$	3, R1	1, R3	0, R3	∞	∞	5, R4	∞	0, R4
$T = 1$	3, R1	1, R3	0, R3	6, R2	12, R2	5, R4	6, R2	0, R4
$T = 2$	3, R1	1, R3	0, R3	6, R2	9, R3	5, R4	6, R2	0, R4

Count-to-Infinity Problem in Distance Vector Routing Protocols

In general, a distance vector routing protocol employs Bellman-Ford algorithm[10] to find the best paths. Since each router running the algorithm builds its routing table based on the update messages advertised by other routers, the information may be inaccurate and lead to routing instability. Furthermore, the typical convergence time of a network employing a distance vector routing protocol is long. This leads to a large amount of packets being dropped when there is a change in topological information. The main cause of the problem is known as *count-to-infinity problem*.

Figure 2.2b illustrates a count-to-infinity problem when link R2—R4 fails. After R2 detects the failure, it has no further knowledge of where to forward the packets to R4 since link R2—R4 becomes unavailable. Therefore, R2 employs the distance vector of its neighbour, R3, which indicates that packet can be forwarded to R4 via itself. Consequently, R2 broadcasts its new distance vector to all of its neighbours. Once R3 and R1 receives this update, they propagate it to inform the rest of the network that there is a path to R4 via R3 with a cost of 7 (*i.e.* the sum of costs from R3 to R4 and R2 to R3). After R3 receives this advertisement, it does not realise that the routing path involves itself; hence, it broadcasts the new distance vector indicating that it can forward packets to R4 via R2 with a cost of 8. This process iterates until the routers realise that R4 is no longer reachable. Count-to-infinity problem usually slows down the convergence process. It also increases the computational and message overheads as well as consuming the network resources.

[10] An algorithm derived from [10] and [34], which is described in [11].

Solutions to Count-to-Infinity Problem

From the operational point of view, network stability and reliability are important features. They are the main factors that determine the QoS of a network. Service disruptions due to unstable and unreliable network may cause the organisations a huge loss in profits. In order to avoid this, the count-to-infinity problem in distance vector routing protocols must be avoided. The followings describe techniques developed in different routing protocols [78].

- **Hop count limit**—is used to indicate the maximum hop count that can be incremented by the routers along the path between each source-destination node pair before it is considered unreachable. For example, the maximum hop count specified in the specification of RIP is 15 [70]. This allows the protocol to avoid routing loops caused by indefinite forwarding. However, this imposes a restriction on the network size (*i.e.* no more than 15 hops away subnetworks are supported).

- **Split horizon**—prevents routing loops by allowing routers to omit any updates they receive from the same interface. Consider an example in Figure 2.2, if R1 has the information of the path R1→R2→R3, it will not propagate the distance vector of R3 to R2. Many distance vector routing protocols such as RIP, IGRP and Enhanced Interior Gateway Routing Protocol (EIGRP)[11] employ this technique to minimise forwarding loops.

- **Poison reverse**—is a technique used in conjunction with split horizon. Split horizon cannot eliminate all possible forwarding loops. Poison reverse sends the routing information along the path backwardly. A combination of split horizon and poison reverse is known as *infinite split horizon*. The technique is effective in term of preventing routing loops as each router sends updates with unreachable hop counts back to the sender for every route it receives. Nevertheless, infinite split horizon requires larger update messages; therefore, it is not commonly implemented in routing protocols.

- **Hold-down timer**—sets the period to suppress a router from receiving further updates after it first receives an information about an unreachable path. If the value of the timer becomes 0, a router indicates the path as being inoperable. A hold-down timer is implemented in routing protocols such as IGRP and Distance Vector Multicast Routing Protocol (DVMRP) [101].

- **Triggered updates**—forces a router to send updates immediately once its routing table changes. It is often used in conjunction with poison reverse to ensure that all routers in the network are notified of the failure before their hold-down timer expires.

In addition to these techniques, a routing protocol such as EIGRP solves the routing loops problem by implementing a strict control over updates between routers [2]. Nevertheless, the implementation of routing protocol becomes more complicated; hence, it slows down the convergence process. As a result, distance vector routing protocols are generally appropriate for deployments in small networks due to their simple implementation and routing policies.

[11] A Cisco's proprietary routing protocol based on IGRP with various optimisations regarding stability, processing power, and bandwidth utilisation.

2.1.2 Link-State Routing

For a network that employs a link-state routing protocol, all routers share the same topological information. That is, the algorithm employs a replicated distributed database approach. When a network forms, each router discovers its local routing information, that is, detects its neighbours and the weights of links used for connections. After that, this information is advertised to the rest of the network in form of protocol control messages. The process is also known as *link-state advertisements*. These advertisements form up a link-state database[12], which is synchronised among all routers. With this topological knowledge, a router can individually perform the shortest paths calculation and build a routing table. Once all routers complete their routing table constructions, the network converges. Examples of shortest path algorithms are Dijkstra's algorithm [27] and Bellman-Ford algorithm. The latter can be used to calculate the shortest paths for network topology that incorporates negative link weights. However, it takes longer to complete the algorithm. More importantly, the costs of paths between any node pairs can be determined using different metrics such as delay and physical distance.

The key features of a link-state routing protocol are: *a)* loop-free path computation and *b)* shorter convergence time compared to that of distance vector routing protocols. However, the convergence time depends heavily on the network size; therefore, it can take longer for a larger network to converge when there are topological changes (*e.g.* node and link failures). Another factor that has impacts on the convergence time is the formation of the topology. An example of a link-state routing based on Dijkstra's algorithm is illustrated in Figure 2.3.

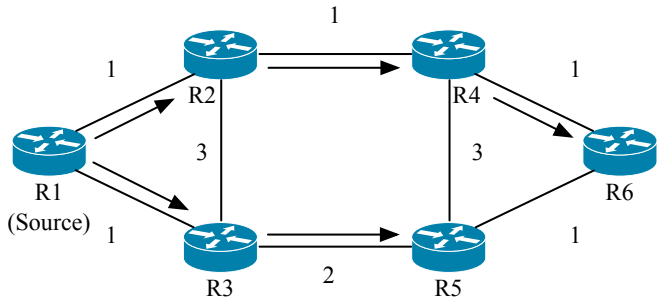


Figure 2.3: Routing based on Dijkstra's algorithm.

Basically, for a given metric, Dijkstra's algorithm computes the shortest path tree rooted at the source node. In Figure 2.3, R1 represents the source node running Dijkstra's algorithm. The shortest path tree rooted at R1 can be found by connecting all arrows illustrated in the figure.

Let T be the number of steps representing the process. The shortest paths computation at R1 is described as follows:

- The algorithm initialises at $T = 0$, where R1 considers the weights of links connecting itself to its immediate neighbours. The routing table is updated for the first time. As only R2 and R3 have

[12] A topological database that stores information on the states of all links in the network

direct connections to R1, the costs of paths to all other nodes in the network are equal to infinity.

- At $T = 1$, R1 chooses the next node to determine based on the current link-state information. Since R2 is connected to R1 with the least cost, R1 updates the information on costs of links connecting R2 to its neighbours. The process aims to find out if there are any better paths available for all destinations. Since R2 has connections with R5 and R6, R1 updates its routing table with corresponding costs and next hops.

- The process iterates until all nodes are considered and R1 possesses a complete link-state information. Table 2.2 shows the steps of routing table construction at R1.

Table 2.2: Routing table construction at R1 based on Dijkstra's algorithm.

T	**Determined nodes**	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**
0	R1	0,R1	1,R1	4, R1	∞	∞	∞
1	R1, R2	0,R1	1, R1	4, R1	2, R2	∞	∞
2	R1, R2, R4	0, R1	1, R1	4, R1	2, R2	∞	3, R2
3	R1, R2, R4, R6	0, R1	1, R1	4, R1	2, R2	6, R2	3, R2
4	R1, R2, R3, R4, R6	0, R1	1, R1	4, R1	2, R2	5, R3	3, R2
5	R1, R2, R3, R4, R5, R6	0, R1	1, R1	4, R1	2, R2	5, R3	3, R2

Designing a routing protocol based on a link-state algorithm offers several advantages. First, it offers a faster convergence time compared to that of distance vector routing protocols. Second, it guarantees a loop-free environment when there are no topological changes. Even if changes do occur, duration of the forwarding loops are very short (*i.e.* transient loops). Third, a synchronised link-state database provides a global information that allows packets to be forwarded along the shorter paths available. Moreover, the overheads of link-state routing protocols are typically small. They can perform well under most routing scenarios and in all practical networks. Nevertheless, they are much more complex than distance vector routing protocols and required skilled network operators to plan, implement, and configure. Examples of widely deployed routing protocols based on link-state algorithms are Open Shortest Path First (OSPF) [90] and Intermediate System to Intermediate System (IS-IS) [54].

2.1.3 Path Vector Routing

As described previously, the Internet interconnects a large number of networks. These networks are generally referred to as Autonomous Systems (ASes). In practice, each AS may consist of a large number of organisations. Nevertheless, many ASes may also form a single ISP. Furthermore, the administration of each AS may be different from one to the others, and often driven by security, efficiency, and fiscal reasons. Consequently, the routing process can be categorised into intra-domain and inter-domain levels. The former category handles the routing within a single domain while the latter deals with routing operations across ASes.

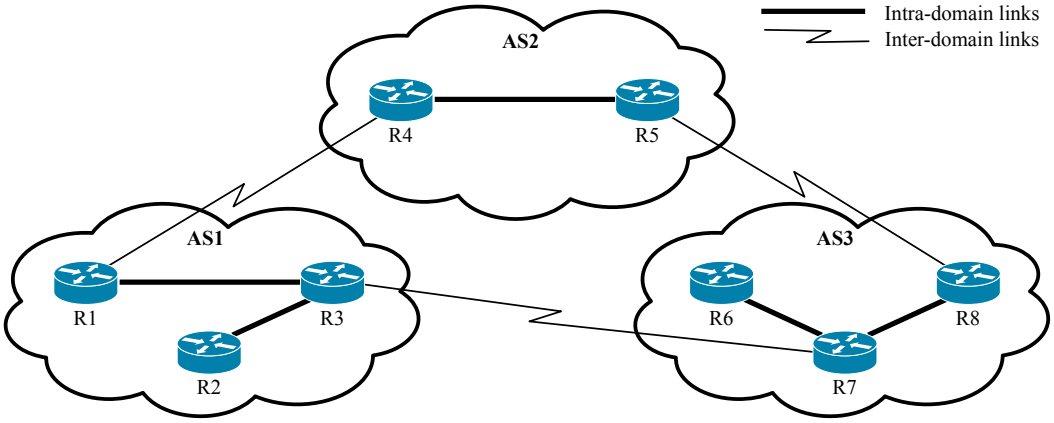


Figure 2.4: Intra-domain and inter-domain routing.

Evidently, distance vector and link-state routing protocols are not appropriate for communications across networks as they become intractable. For example, RIP introduces routing instability in larger networks while OSPF requires a large amount of computational power and memory to store routing tables. Unlike RIP and OSPF, the Border Gateway Protcol (BGP) [103] is widely used for inter-domain routing. It is considered as a path vector routing protocol [32].

Path vector routing protocol is often regarded as a class of distance vector routing protocols due to similarity in broadcasting the local routing information of a router. However, a path vector routing protocol employs various path properties as its metric. These path attributes and the destination together define a path vector. A router running a path vector routing protocol prevents the routing loops by defining a special path attribute that records the sequence of the nodes based on the information on reachability received from its neighbours.

Figure 2.5 illustrates an example of path advertisements. Let AS4 be the destination, the routing information towards AS4 is obtained as follows.

- AS4 sends the path advertisement to AS2. Thus, AS2 has knowledge of the path AS2→AS4.
- AS2 propagates the path advertisement it receives from AS4 to AS1 and AS3. Basically, AS2 has the ultimate decision whether it wants to advertise this path.
- AS1 and AS3 are notified of the path to AS4 via AS2 and both of them propagates the advertisement towards each other.
- The path advertisements from AS1 to AS3 and AS3 to AS1 are ignored.

In practice, if AS2 decides not to advertise the path to AS3, it might receive a path advertisement from AS3 which incorporates itself. However, since the information of all nodes along the path is embedded, the advertisement will be discarded.

As path vector routing protocols do not require all nodes to have a homogeneous routing policy, each AS may employ different policies and path selection process. Figure 2.6 illustrates local policies

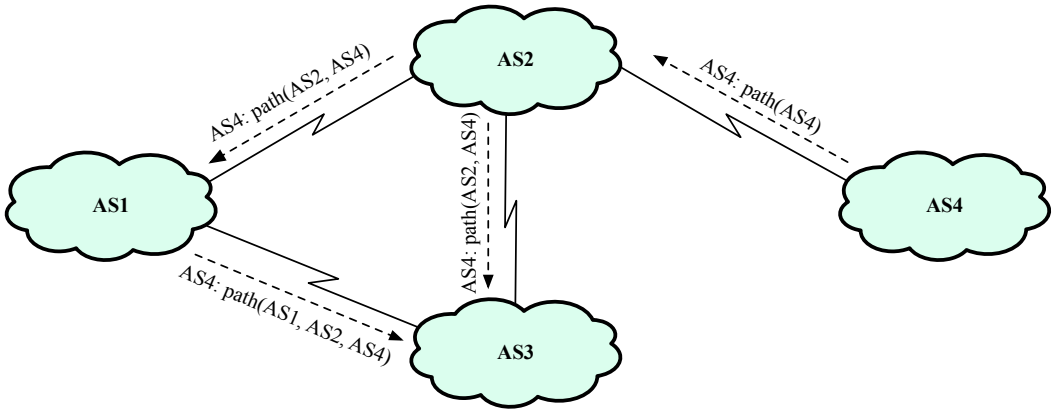
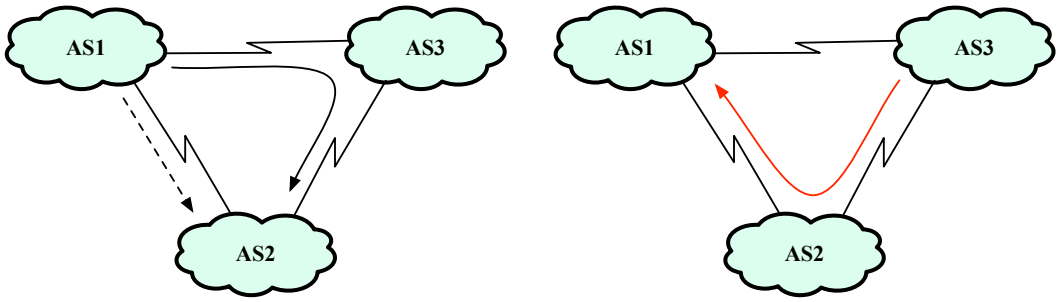


Figure 2.5: Path vector routing.

applied by different ASes. In Figure 2.6a, the shortest path from AS1 to AS2 is direct, but AS1 prefers to send packets to AS2 via AS3. There are many potential reasons for AS1 to select a longer path given that the number of hops is used as a metric in this example. For example, the path via AS3 may consist of high-speed and/or higher capacity links. Figure 2.6b shows that although AS2 has a path towards AS1, it does not prefer to advertise this route to AS3.



(a) Path preference.

(b) Hidden path.

Figure 2.6: Different network policies in path vector routing.

Although routing protocols are categorised by their routing algorithms, additional mechanisms may be employed to provide distinctive features. The following section describes and compare the well-known intra-domain and inter-domain routing protocols.

2.2 Traditional Routing Protocols

2.2.1 Intra-Domain Routing Protocols

Routing within a single AS requires an intra-domain routing protocol known as Interior Gateway Protocol (IGP). The protocol defines the means for routing table construction and routing information exchange between routers in the same network.

Routing Information Protocol

In the early days of the Internet, communications within a single domain was performed through the use of RIP. The first version of the protocol was originally defined in RFC 1058 [47] solely for exchanging routing information between routers. Due to its limitations, the protocol has been extended with various mechanisms resulting in its second version specified in RFC 2453 [70].

RIP is mostly deployed in moderate size ASes. Due to its simple implementation, it is not supposed to be used in complex networks. In general, the protocol employs Bellman-Ford algorithm. Practically, a router employing RIP exchanges routing information with its neighbours. Thus, the packets destined for routers that are not directly reachable need to be forwarded to one of the router's immediate neighbours (*i.e.* next hop router). The selection process is performed based on the routing information available at a local router. That is, the neighbour that offers a better path to the destination is selected as the next hop router. Under RIP, the shortest path determined by its physical distance does not essentially imply the best path. RIP selects the path that has the least number of hops. It also requires that the network must be contiguous[13] in order to employ the protocol.

Under the first version of RIP, each router initialises the protocol by determining the status of all of its interfaces (*i.e.* the connections to its neighbours). After that, it enters a request-full mode by broadcasting a request message to its neighbours. As the neighbour nodes receive this message, they respond to the originating router following the split horizon rules and poison reverse technique [47, 70]. A router updates its routing table only if it receives responses from its neighbours successfully [78].

In practice, multiple paths with equal costs may be available for some destinations in the network. The implementation of RIP conforms to the basic specification does not employ any rules about Equal-Cost Multi-Paths (ECMP)[14] for load balancing. For example, if there are four paths to the same destination with equal costs, only the first one is learned by the router. However, some proprietary implementations also allow ECMP routing under RIP [123].

In order to ensure accurate routing information, routers running RIP perform updates periodically. By default, RIP routers send updates to their neighbours every 30 seconds. This interval is set and triggered by an auto-update timer[15] [47]. In addition, the protocol also uses triggered updates to prevent routing instability [47, 70].

RIP handles changes in topological information resulted from removal or failures of network elements by setting a time-out for a route towards each existing destination. If a router does not receive any response from its neighbour after sending a request message for more than 180 seconds, it considers that the route has expired and starts the route deletion process [78].

Despite the fact that some networks are still employing RIPv1 as their routing protocol, various problems have been found. From an operational point of view, RIPv1 can perform well in small to moderate size networks given that the networks are reliable (*i.e.* routers and links are very reliable). However, if there are unexpected changes in the topology, it may incur a vast amount of losses [78].

[13] Networks and routes must have the same class network address.

[14] A routing scheme that allows packet forwarding towards the same destination over multiple best paths.

[15] Note that, the timer is not affected by the load in the network.

The second version of RIP was introduced with additional capabilities such as support for authentication and subnet masking. These mechanisms have become available through the modifications of the format of protocol message. Several unused fields in RIPv1 messages are employed by RIPv2 to provide additional routing information. As RIPv2 also supports non-broadcasting network, multicasting the messages is more common than broadcasting them as in RIPv1.

Despite its simple implementation, RIP has some limitations. Although precautions have been taken in order to solve some obvious routing problems, they do not cover all possible cases as RIP was originally designed for routing in moderate size networks. The details of these limitations are described in RFC 2453 [70].

- It cannot be employed in the network with a path longer than 15 hops. RIP metric is restricted to an integer ranging from 1 to 15 inclusive. This prohibits the protocol from being incrementally deployed. That is, the protocol cannot be used in large networks.

- In certain failure scenarios, RIP cannot avoid the count-to-infinity problem. This becomes a practical concern in large networks as the update messages can consume a huge amount of bandwidth. However, these cases are considered fairly unusual.

- RIP employs a fixed metric, which is the hop count. Thus, it lacks of flexibility when real-time parameters such as delay and traffic load become important for path selection. According to RFC 2453, some extensions may be implemented to permit the usage of other metrics. However, these extensions potentially introduce routing instabilities.

These limitations restrict the RIP extensions and RIP-based protocols from further development. Furthermore, RIP-based protocols cannot violate these constraints due to the need for backward compatibility.

As the Internet evolves, a routing protocol must adapt to new environments. Since the original RIP does not address any security issues, RIPv2 employs a cryptographic authentication [6] to prevent passive attacks in the Internet. Despite its slow deployment [44, 28], the emergence of IPv6[16] demands compatibility from existing routing protocols. Thus, RIPng [72] has been proposed to define the minimum changes to RIPv2 specification for IPv6 compatibility.

Although RIPng is based on RIPv2 and employs similar routing algorithm, which incurs a slow convergence process, it still has its place for deployment. More precisely, it can be used in small networks due to its simple implementation and administration. Regardless of its support for IPv6, RIPng inherits the same limitations from the original RIP.

In addition, it is important to realise that the main problems arising in RIP are routing loops which often lead to routing instability due to unsynchronised routing information. Consequently, a new routing protocol based on the original RIP called Routing Information Protocol with Minimal Topology Information (RIP-MTI) [118] has been proposed. Under RIP-MTI, routing loops are avoided as the routers

[16] Next Generation Internet protocol specified in RFC 2460 [25] as a replacement of IPv4 [98] to increase flexibility in allocating addresses and eliminating the need for Network Address Translation (NAT).

recognise and reject the false updates. In other words, any updates that may lead to a count-to-infinity problem will not be propagated. The router employing RIP-MTI recognises these falsified routes by evaluating a simple metric-based equations [108]. Furthermore, there have been several RIP extensions proposed in order to deploy RIP in new environments. For example, RFC 1724 [71] specifies a Management Information Base (MIB)[17] to manage RIPv2 devices. Another example is RFC 1582 [83], which defines extensions to RIP to support demand circuits.

It is greatly believed that RIP-based protocols such as RIPv2 and RIPng will remain for sometime in the Internet regardless the success of link-state protocols such as OSPF and IS-IS.

Open Shortest Path First

OSPF is an IGP designed specifically for routing in TCP/IP networks[18]. The most widely deployed OSPF in current networks is OSPF version 2 (OSPFv2) specified in RFC 2328 [90].

OSPF is perhaps, the most successful dynamic routing protocol deployed in enterprise networks. It employs a link-state algorithm to construct routing tables. More precisely, it uses Dijkstra's algorithm to compute the shortest paths. OSPF requires complete information on the network topology. Unlike RIP, OSPF supports the use of ECMP through modifications of the original Dijkstra's algorithm to permit identification of equal-cost paths. When two paths with equal-cost exist, the next hops of both paths are stored in the routing table. The router can spread the load among these paths to reduce congestion [78].

Basically, the protocol depends heavily on the Link-State Advertisements (LSAs) originated by each router in the network. They are generated based on the local routing information of their originating routers. LSAs are generally flooded using a reliable method [91], which is regarded as an in-network functionality [50]. The process allows each router to exchange routing information with other routers by originating, updating, requesting, and acknowledging to LSAs. In other words, an OSPF router constructs its routing table based on information embedded in the advertised messages it receives. A collection of LSAs also allows the router to build a link-state database, which is synchronised with all other routers. This permits a consistent routing information as long as the network is stable (*i.e.* routing may be inconsistent during network re-convergence).

It is important to note that, in practice, the network is often subject to changes. For example, a link or a router may be added or removed. Thus, it becomes vital for a router to discover and maintain the relationships with its neighbours. In an OSPF network, these processes are performed by the Hello protocol, which is a part of the protocol suite. Each router simply originates and sends a Hello packet to all its interfaces (neighbours) periodically to inform of its existence. The process also allows routers to detect local failures caused by failed links or inoperable neighbouring routers. Nevertheless, it cannot indicate the root cause of the failure. As a result, the re-calculation of the shortest paths in OSPF always assumes a link failure. Furthermore, the Hello protocol must be employed consistently throughout the network to avoid falsified decision on maintaining neighbours.

Basically, OSPF protocol runs directly over the IP network layer. Thus, OSPF packets are always

[17] A database used to manage devices in communication networks.

[18] Networks that employ Internet protocol suite.

encapsulated in IP packet headers. It also relies on IP mechanisms to fragment the packets when their sizes exceed the MTU. However, RFC 2328 suggests that IP fragmentation should be avoided whenever possible as it may cause the loss of functionality in transferring large packet types. Consequently, large OSPF packets are often split into several protocol packets.

In general, the networks forming the Internet may have different infrastructure. More precisely, they employ different data link technologies such as frame relay, token rings, Fibre Distributed Data Interface (FDDI) rings, and Asynchronous Transfer Mode (ATM). OSPF handles various network types differently. The protocol can perform subnetwork dependent tasks, which include: *a)* neighbour discovery and maintenance; *b)* database synchronisation; and *c)* abstraction.

From an operational perspective, routing protocols should support hierarchical routing as it is an essential technique commonly used in large networks. The notion behind this requirement is mainly due to the fact that the complexity of administration and control of the network is directly proportional to the network size. In other words, more resources and functions are required for larger networks [78].

OSPF supports hierarchical routing by dividing a single network into several areas known as *OSPF areas*. Basically, these areas are assigned with ID numbers (stored in Area ID field in OSPF packet). The topological information of the network is not flooded across area; that is, no router-LSAs or network-LSAs are sent across the area borders. Nevertheless, OSPF allows routers to be attached to two or more areas known as Area Border Routers (ABRs), to discover external destinations. The information on addresses available through the ABRs is described in summary-LSAs and generally determined by internal routers during the path selection process [91]. In addition, OSPF also supports virtual links to provide logical attachments between OSPF and the backbone areas. Despite this logical information, virtual links are not incorporated in the shortest paths calculation.

Routing protocols are often demanded for compatibility with the upcoming technologies, or to provide additional features as requested by network operators due to various reasons. For example, an ISP may want to provide certain QoS as agreed with its customers. Similar to RIP and other protocols, OSPF has been developed to support IPv6 (*i.e.* OSPF version 3 [24]). Due to the modifications in data format, additional capabilities and extensions to OSPFv3 are possible. Based on RFC 2328 [90], OSPFv2 has already offered two optional capabilities, which are TOS-based routing and support for routing in stub areas. Nevertheless, the extensions must provide backward compatibility with older versions of the protocol. Extensions and capabilities added to OSPF include TOS-based routing[19], QoS-based routing[20], Stub area, and Not-So-Stubby-Area (NSSA) [92].

Since an AS needs to interact with other ASes, the routing information of an inter-domain routing protocol must be transmitted across ASes. Basically, this operation is handled by the Internal BGP (IBGP) [103]. However, OSPF also provides similar mechanisms using the external-attributes-LSA to import all AS paths into the routing domain [91].

Recently, multicast has become a requirement for several applications including multicast radio[21].

[19] TOS-based routing is dropped in IPv6 [25].

[20] QoS-based routing is an experimental extension[5].

[21] BBC provides multicast radio to enhance the audio quality over the Internet (`http://www.bbc.co.uk/multicast/`).

Multicast extensions to OSPF (MOSPF) [89] has been proposed to support multicast in OSPF networks by introducing a group-membership-LSAs without altering other packet formats.

As the Internet grows, routing within a network becomes more complex due to unpredictable traffic patterns and unreliable network resources. Traffic Engineering (TE) [58] and Multi-Topology (MT) [100] extensions have been proposed as solutions to enhance the network performance. In addition, near optimal traffic engineering solutions for current OSPF networks without requiring any major modifications to the routing standard have been proposed [116].

Although TOS-based routing is deprecated, MT-OSPF reflects its capabilities in many ways. A number of routing trees are computed based on different criteria and classes of service. Nevertheless, MT-OSPF allows individual links or prefixes to be excluded from the topology [100].

Despite the fact that RIP and OSPF are well-known routing protocols employed in numerous ASes, there have been other intra-domain routing protocols developed such as Cisco's proprietary routing protocols [48, 33, 36] and IS-IS[22] [54, 19], which is an international standard within the Open Systems Interconnection (OSI) reference design. As these protocols are based on either distance vector or link-state algorithms, the detailed descriptions of these protocols are omitted.

2.2.2 Inter-Domain Routing Protocols

In practice, it is not scalable to employ protocols such as OSPF and IS-IS for routing across ASes. As the Internet continues to grow, it is difficult (if not impossible) for an AS to acquire the global topological knowledge of all interconnections. Thus, routing packets across ASes requires different routing protocols known as Exterior Gateway Protocols (EGPs).

Exterior Gateway Protocol

The name EGP is often confused with a generic term used to describe inter-domain routing protocols. It is an obsolete protocol used in the early Internet for communications between ASes. However, the impact of this protocol on modern routings still remains, in particular, its concept of dividing the Internet into different ASes [50].

In general, routing information is stored in routing tables. However, this information does not propagate outside the network. Instead, it is stored at local routers based on the IGP. Consequently, internal routers have no knowledge of the external routes. EGP aims to permit routing between ASes through exchanging information on the external destination prefixes each AS can reach. Basically, the protocol needs to determine its neighbours, which must be agreed by both ASes. If two ASes become neighbours, EGP is responsible for monitoring and maintaining the connection. These functions are similar to IGP's neighbour discovery and maintenance processes. Nevertheless, EGP is not a link-state routing protocol; hence, each AS does not have the complete information on the Internet topology. As a result, it needs to exchange the routing information with its existing neighbours [50].

It is important to note that, routing in the AS level needs to be agreed by both neighbours. That is, even if there is a physical link connecting two ASes, it might not be employed without neighbour

[22] Additional capabilities proposed for IS-IS have been continuously developed to support routing in IP networks. These can be found at: `http://www.ietf.org/html.charters/isis-charter.html`.

acquisition. Furthermore, an AS may not expose some of its routing information to certain neighbours due to its routing policy. In practice, each AS running EGP advertises its reachability list computed based on the internal routing tables constructed by IGPs such as OSPF and IS-IS. EGP routers perform the path computation based on the metric. However, this metric is different from those in IGPs as it does not signify any consistency. Consequently, it uses 255 to identify unreachable destinations and lower integers to reflect the path's preference [105]. EGP provides a simple method for constructing the routing tables. After an AS agrees to become neighbours with its adjacent ASes, it determines whether all neighbours have advertised their reachable destinations (lists of reachability). Once the lists are received, the AS compares and decides the best path for each reachable destination [105, 50].

Since EGP does not provide a complete routing information, messages exchanging between ASes may result in falsified paths, if at least one or more exterior gateways are wrongly configured [105, 50]. Furthermore, as the protocol was designed to handle a simple tree-like topology, employing EGP in a more complex network may result in routing loops [109]. As EGP embeds all information on network reachability in a single IP packet, the size of the message often exceeds the MTU. This leads to packet fragmentation. However, this problem can be solved by adding certain mechanism for fragmentation[23] [50].

Border Gateway Protocol

Recently, the growth of the Internet has been driven by many organisations both in business and academic sectors. This results in a very complicated global topology. Consequently, communications between interconnected networks need a better routing protocol (rather than EGP) with enriched features and appropriate mechanisms to support various operations. At last, BGP was created to replace EGP and has been widely deployed for decades.

Similar to EGP, BGP is a routing protocol used to exchange the information on the network between ASes. In general, for two ASes to communicate, they are required to set up a session using Transmission Control Protocol (TCP) known as TCP session, in order to ensure a reliable delivery [78]. This session stays connected to maintain the neighbour relationship and update information periodically.

BGP makes use of the Classless Inter-Domain Routing (CIDR) concept. Basically, the network numbers are divided into 3 classes, A (24 bits addressing), B (16 bits addressing), and C (8 bits addressing). However, each Class C address can support only 256 hosts. Most organisations such as universities and large companies have more than 256 Internet-capable devices; thus, Class C addresses become insufficient. In contrast, a single Class A address can support up to 2^{24} hosts. Nevertheless, the addresses are very limited and not easily given by the Internet Assigned Numbers Authority (IANA). As a result, most organisations prefer Class B addresses, which are best options to them. Since the Internet is growing at a rapid pace, Class B addresses are being depleted [51]. Furthermore, the size of routing tables grows proportionally to the number of interconnected networks; hence, the growing Internet incurs larger memory requirement. CIDR solves these problems by allocating subsets of classful addresses (*i.e.* Class A, B, and C addresses). Each IP address under CIDR consists of the prefix, which identifies the

[23] This has not been done due to the emergence of BGP.

network, and the host address of particular network prefix. CIDR is based on Variable-Length Subnet Masking (VLSM) to increase the flexibility of address allocation [78, 119]. Moreover, CIDR allows *route aggregation*[24] to ensure that the routing table size is not too large [50].

BGP employs the conceptual model of path vector. The message is generally embedded with information on every node that forms a path. This prevents BGP from routing loops incurred in distance vector routing protocols. BGP refers the routing table as the Routing Information Base (RIB) of a local router. It also specifies the terms, Adj-RIB-In, Adj-RIB-Out, and Loc-RIB to describe RIBs that store learned prefixes, advertised prefixes, and selected prefixes, respectively [119].

One of the most important features of BGP is the use of path attributes. These attributes are used to inform ASes of information on the prefixes of a path. The formal specification of BGP can be found in RFC 4271 [103].

Basically, BGP is divided into Internal BGP (IBGP) and External BGP (EBGP). EBGP is used to communicate between ASes. However, when an AS learns the prefixes from its neighbours, it needs to distribute this information within the network. This process is generally performed using IBGP [78].

Similar to other protocols, after each AS learns external prefixes from other ASes, it needs to select a path for each destination prefix. In general, the path with the highest LOCAL_PREF will be used. If there are more than one routes with the same LOCAL_PREF, the shortest AS_PATH is selected (*i.e.* the path with a minimum number of hops). Nevertheless, if the MULTI_EXIT_DISC attribute is considered by the AS, the route with the lowest MULTI_EXIT_DISC is selected. If there are equal MULTI_EXIT_DISC paths, the AS needs to determine the NEXT_HOP attribute and selects the route that has a minimum cost. Nevertheless, if the routes are learned via EBGP/IBGP, the path through EBGP/IBGP with the lowest identifier is selected instead [119].

In general, customers connect to an ISP via single connections. However, they can practically subscribe for multi-homed sessions to permit load balancing between two links. One of the main reasons is to share the load between multi-homing paths, which increases the reliability of the communications.

Fiscal model is an important factor that allows the ISPs to survive in the industry. Routing traffic consumes bandwidth; therefore, different service providers must be able to employ their own routing policies. For example, an ISP may be willing to act as a transit network for another ISP due to some business agreements. However, as a profit driven organisation, it is unlikely for an ISP to provide a transit without gaining any return. BGP allows ISPs to manage their own routing policies. In practice, however, ISPs often filter the advertisements of prefixes to prevent themselves from acting as transit networks (*i.e.* the destinations of the traffic flows are not in the network). Furthermore, advertising or forwarding the prefixes to the neighbour ASes needs to be selective to prevent any undesirable results [119].

BGP has several extensions since it was first specified to be in line with the dynamic behaviours of the internet. These include features such as route flap dampening, management of complex routing policies, TCP MD5 authentication, support for Virtual Private Networks (VPNs), and support for backward compatibility [119, 22, 104].

[24] A group of network numbers is represented by a single routing entry.

Although BGP is widely deployed, configuring the routing tables of BGP nodes is difficult and needs to be performed by a skilled operator. It is greatly believed that, before IPv6 dominates the Internet, new parameters will be added to BGP to cope with the growth of the interconnections.

2.3 Resilient Routing

Internet routing has created many challenges since the beginning of the era. During the eighties, data communications via the Internet were not widely deployed. Many researchers believed that the number of global interconnecting computers would be much less than in the present days. Thus, it is not surprising that the original design of the Internet, including its legacy infrastructure will fail to accommodate the upcoming services and applications. Although the state of art in networking hardware offers significantly higher performance and reliability than in the past, it does not guarantee a 100% uptime. In addition, the network reliability cannot be measured alone by the performance of network equipment. Other factors such as human errors in configuring the network and intentional attacks (*e.g.* fibre cuts) must also be considered.

Even though faults caused by human errors and attacks are unavoidable, routing protocols or certain mechanisms should be employed to guarantee a successful delivery of packets. Otherwise, one would be anxious to subscribe for Internet-based services that require highly reliable networks.

The rest of this chapter presents the existing approaches proposed for handling unexpected failures. Although some techniques cannot guarantee a 100% successful delivery due to unrecoverable failures, they can be used to enhance the performance of a network to some extent. Before exploring any resilient techniques, it is important to understand the goals of these solutions.

2.3.1 Design Goals

It is practically important to design a protocol or technique that can be deployed in real networks. This section concisely describes the framework and goals of network reliability solutions. The following terms are defined to ensure consistent and accurate expositions.

Definition 2.1 (Reliability)**.** *Reliability is defined as the ability of a network in handling failures or systematic attacks without impacting network operations.*

Definition 2.2 (Completeness)**.** *The solution to the reliability problem is complete if it guarantees a successful packet delivery in case of network failures. However, the completeness is bounded by the aim of resilient mechanisms. If, for example, the main objective of a routing strategy is to handle single link failures, the completeness is achievable when all single link failure cases are taken into account. The completeness can be measured as a percentage of the repair coverage described later in this section.*

Definition 2.3 (Correctness)**.** *The solution to the reliability problem is correct if, in the presence of failures, it can deliver the packet successfully without creating continuous forwarding loops.*

If the mean of a solution to the reliability problem is to provide an alternate path for packet delivery, the terms alternate path, path length stretch, and repair coverage are defined as follows.

Definition 2.4 (Normal path)**.** *The normal path is defined as the path used in normal operation (i.e. in the absence of failures). Although it is commonly used, the normal path is not restricted to the shortest path. Furthermore, the terms "normal" and "primary" are used interchangeably.*

Definition 2.5 (Alternate path or backup path)**.** *The alternate path also known as the backup path is defined as the path employed by routers to deliver packets when the optimal shortest path becomes unavailable due to network failures. Furthermore, the terms "backup", "alternate", and "secondary" are used interchangeably.*

Definition 2.6 (Path length stretch)**.** *The path length stretch or* stretch *is defined as the excess latency or weight or number of hops required for delivery via the alternate path. It is represented as the ratio of the latency or weight of the alternate path over the optimal shortest path. If two shortest equal-cost paths exist, the stretch of its alternate is 1.*

Definition 2.7 (Repair coverage)**.** *The repair coverage is defined as the percentage of network elements which can be fully protected for all destinations, or the percentage of destinations which can be fully protected for any network elements, or the percentage of the total potential failure cases which are protected.*

Typically, routing techniques and other resilient mechanisms impose specific requirements. In addition, certain schemes have impacts on normal operations under failure scenarios. The following metrics are commonly used in evaluation of resilient strategies.

Definition 2.8 (Overheads)**.** *The overheads are defined as the requirements for deploying certain solutions. These include:* a) *computational overhead (milli-seconds);* b) *memory overhead (number of additional routing table entries);* c) *packet or message overhead (bits or bytes); and* d) *processing overhead (milli-seconds).*

Definition 2.9 (Link Load)**.** *The link load is defined as the amount of traffic passing through a network link per second (Mbps).*

Definition 2.10 (Maximum Link Utilisation)**.** *The Maximum Link Utilisation (MLU) is defined as the maximum ratio of the link load over its capacity. If the link is fully utilised, the link utilisation is 1. However, if the link is overloaded, its MLU exceeds 1.*

Definition 2.11 (Network Overhead)**.** *The network overhead is defined as the sum of loads on all links in the network (Mbps). It is typically increased when a failure occurs and the traffic needs to be delivered through different paths.*

High Reliability

The reliability of a network can be measured using different metrics described previously. In practice, there are several ways to ensure successful packet delivery. For example, any oriented connection such as TCP/IP connection guarantees that packets are sent and received correctly, as both end points of the communications acknowledge each other. That is, when a failure occurs along the path the packets are

being sent, the packets do not reach the receiver. Consequently, the sender notices this unsuccessful delivery as no acknowledgements are received. Therefore, it originates re-sends the packets to the destination. Nevertheless, this mechanism incurs some delays before the packets are actually delivered. Although TCP yields a reliable communications between two parties, packets are still being dropped in the presence of failures.

The idea of a highly reliable network is to ensure that packets are delivered successfully with a minimal delay when there is a network disruption. This can be done by employing routing strategy that guarantees successful packet delivery even if a failure occurs on the forwarding path.

Minimal Requirements

Although a highly reliable network is required, routing techniques cannot be designed arbitrarily due to the limitations of legacy protocols and infrastructure. For example, routing in the traditional IP networks allows routers to perform forwarding based only on the destination IP address. Moreover, not all network equipments have huge resources (*e.g.* memory and processors); thus, one of the design goals is to keep modifications to the existing standards to a minimal.

Minimal Impacts on the Network

The main objective of resilient mechanisms is to enhance the reliability of a network by handling failures efficiently and as fast as possible. However, it is desirable that it has minimal impacts on the network. For example, if a different path is employed when a failure is detected, forwarding along that path must not increase the traffic congestion in other parts of the network significantly (*i.e.* it must not increase the MLU or the total network overhead too severely).

2.3.2 Modifying the Convergence Process

The routing process in the Internet is basically performed by routers based on the routing information exchange. This information is used to construct routing tables, which routers employ to aid packet forwarding. However, when there is one or more changes in the network topology, routers need to update their routing information as well as their routing tables. This is known as the *convergence process*. Packets destined for affected destinations are dropped during this period.

Evidently, before the network starts to converge, a router needs to detect the failure. The time required to detect a failure in a single domain running a protocol such as OSPF is directly proportional to the HelloInterval[25] configured by network operators. More precisely, RouterDeadInterval[26] specifies the time interval a neighbour router can be silent until the router considers that it fails. By default, the HelloInterval and RouterDeadInterval are set to 10 seconds and 40 seconds respectively[27]. Thus, it is possible to speed up the failure detecting time by reducing the HelloInterval. Nevertheless, too short HelloInterval may lead to a falsified information on dead routers or unnecessary convergence process due to intermittent failures. The latter result often leads to routing instability. Furthermore, short HelloInterval can potentially cause network congestion due to a significant higher amount of Hello packets.

[25] A time interval used to discover and maintain the neighbour relationship in OSPF and IS-IS networks.

[26] Usually is set as multiples of HelloInterval.

[27] These figures may vary depending on different implementations.

There is an investigation showing that tweaking the HelloInterval can result in a faster failure detection in OSPF networks [39]. However, the optimal value of HelloInterval depends on the congestion levels and the network formation (*i.e.* the optimal value of HelloInterval heavily depends on the topology). Moreover, this investigation also recommends that the HelloInterval can be much lower than the default value, but should not reduce to the milli-seconds range. Thus, network operators must observe the traffic characteristics and the topological formation in order to find the optimal value of HelloInterval.

In other protocols such as IS-IS, a neighbour router is considered failed when three expected Hello packets are missing. However, since the specification of IS-IS does not allow HelloInterval less than the order of seconds [54], the fastest detection time is 3 seconds [1]. An approach for converging an IS-IS network in order of milli-seconds has been proposed [1]. It recommends the followings for a milli-seconds convergence:

- **Replacing the Dijkstra's algorithm**—to provide a faster Shortest Path First (SPF) calculation. For example, the dynamic Shortest Path Tree (SPT) algorithm can be up to 10,000 times faster than the implementation of Dijkstra's algorithm [1, 128].
- **Modifying the granularity of HelloInterval**—to permit milli-seconds detection as the current specification restricts the failure detection to the range of seconds.
- **Employing distinctive detection schemes**—to differentiate the failure and recovery events.
- **Prioritising Link-State Protocol Data Unit (LSP) propagation**—to ensure that LSPs are propagated before other operations are performed (*i.e.* SPF computation).
- **Prioritising Hello packets**—to allow routers to process Hello packets prior to data packets.

It is important to note that, during the convergence process, not only packets are dropped at the point of failure, but micro-loops are also created due to routing inconsistencies. Consequently, a framework for loop-free convergence has been proposed to alleviate this problem [112]. The framework specifies four strategies to control micro-loops as follows:

- **Micro-loop mitigation**—can be done through fast convergence. In general, the micro-loop duration is proportional to the convergence time; hence, if a network can achieve faster convergence, the micro-loop lasts shorter. Alternatively, a mechanism called Path Locking with Safe-Neighbours (PLSN) [134] may be applied.
- **Micro-loop prevention**—can be done using optional methods: incremental cost advertisement, near-side tunnelling, far-side tunnelling, distributed tunnels, packet marking, new Multi-Protocol Label Switching (MPLS) labels, ordered Forwarding Information Base (oFIB) update, and synchronised Forwarding Information Base (FIB) update [112]. More precisely, oFIB is one of the most common techniques used in a network running link-state routing protocols to prevent transient loops [35]. The mechanism focuses on computing the rank of FIB updates, and is performed sequentially to avoid routing inconsistencies between routers.

- **Micro-loop suppression**—can be done if the routers recognise that the packet is caught in a loop. However, this does not work in an asymmetric network where multiple failures can create cyclic forwarding loops [112].

- **Minimising micro-loops**—can be done through network design. That is, an appropriate network design can reduce the amount of micro-loops due to the topological characteristics.

As described above, tweaking protocol parameters and modifying the traditional convergence process can potentially reduce the routing loops and packet loss rates. Nevertheless, these techniques do not guarantee a successful delivery of the original packets being sent via the failed paths.

2.3.3 IP Fast Re-Route

An analysis in the IP Fast Re-Route (IPFRR) framework [113] shows that, the packet forwarding process can be disrupted when there is a change in the network, in particular, a failure. When a failure occurs, the time typically taken before the network can resume its normal operations is described as follows:

- **Failure detection time**—is the time elapsed from the occurrence of physical failure to the realisation of the failure-adjacent router.

- **Failure reaction time**—is the time required for the detecting router to react to the failure.

- **Message propagation time**—is the time required to inform all routers of the failure.

- **Paths re-computation time**—is the time required to re-compute the new paths.

- **Implementation time**—is the time required to install new path information into the forwarding table.

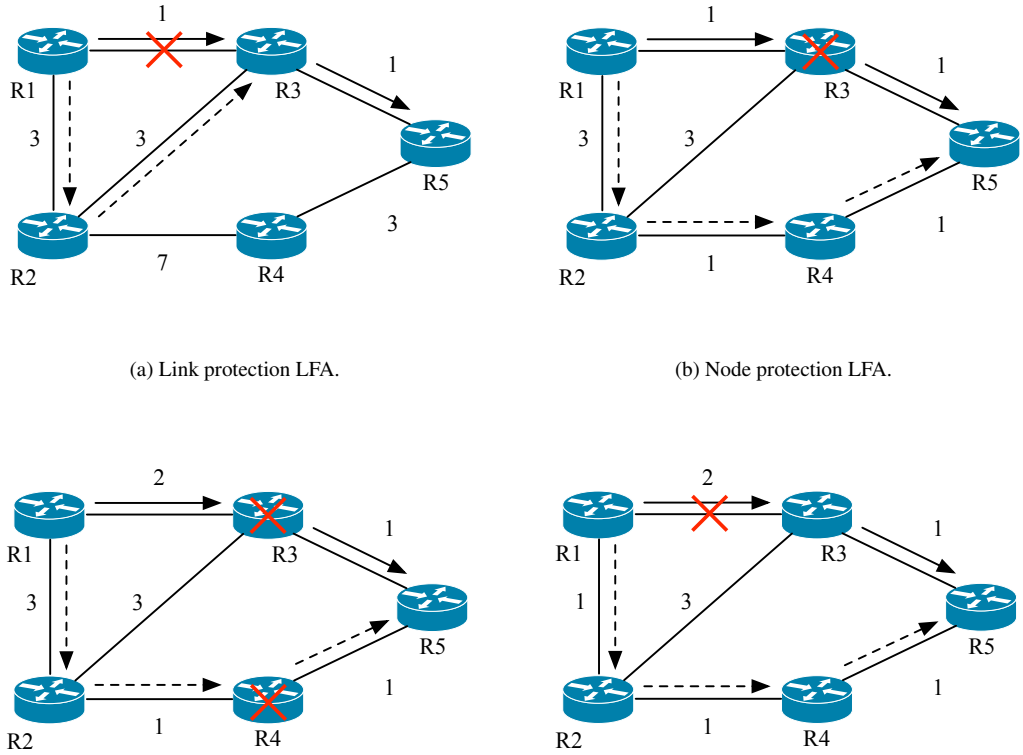
IPFRR specifies two mechanisms to reduce the recovery time, which are: *a)* mechanisms to reduce the failure detection time and *b)* mechanisms to provide repair paths [113].

The failure detection time can be reduced by tweaking protocol parameters as described in Section 2.3.2. This section focuses on different techniques for computing repair paths. Existing fast re-route strategies are now described.

Loop-Free Alternates

Loop-Free Alternates (LFAs) [8] are the simplest techniques used to provide fast re-route. No significant modifications are necessary for deployment. Typically, each routing table stores the information about the next hop for each destination. Evidently, there may exist more than one path between any node pairs. Nevertheless, it is difficult to employ these paths consistently as they can potentially create forwarding loops. Thus, only neighbours that do not cause a packet to traverse back to the point of failure can be used as alternate next hops. In general, LFAs can be categorised based on their ability to serve as alternate next hops. Some LFAs can protect node failures while the others can protect only link failures.

- **Loop-Free Condition**—a neighbour node that does not create any forwarding loops can be used as an LFA. It can be used to re-route packets in the presence of a local link (*i.e.* the link connecting



(a) Link protection LFA.

(b) Node protection LFA.

(c) Downstream LFA.

(d) Equal-Cost Multi-Paths.

Figure 2.7: Different types of LFAs.

between the detecting node and the normal next hop) failure. Let s be the detecting node, d be the destination, and n_p be the normal next hop. The cost of a path between any two nodes, i and j, is denoted as $cost(i, j)$. A neighbour n_i can be used as an LFA to protect link failure if the following condition is satisfied:

$$cost(n_i, d) < cost(n_i, s) + cost(s, d) \tag{2.1}$$

The above condition is known as Loop-Free Condition (LFC), and the neighbours that satisfy LFC are the most basic LFAs used for link protection. Figure 2.7a illustrates a link protection LFA from R1 to R5.

- **Node-Protection Condition**—in practice, the root cause of failures are not limited to link failures, but also node failures. Thus, packet delivery via an LFA that satisfies LFC cannot be guaranteed. Consequently, an LFA must meet the following condition to provide node protection.

$$cost(n_i, d) < cost(n_i, n_p + cost(n_p, d) \tag{2.2}$$

Figure 2.7b shows an example of node protecting LFA. It is important to note that, all node-protecting LFAs also protect link failures while link-protecting LFAs do not guarantee recovery in case of node failures.

- **Downstream Condition**—link-protecting and node-protecting LFAs can be employed to provide re-route paths in the presence of single failures. However, certain cases of multiple failures can create forwarding loops. Consider Figure 2.7b, R2 is an eligible node-protecting LFA for R1 to R5 and R1 is an eligible node-protecting LFA for R2 to R5. Therefore, when the link R1→R3 fails, R1 re-routes packets via R2. Similarly, if R2→R4 fails, packets are re-routed via R1. If both scenarios occur at the same time, packets are eventually caught in a loop between R1 and R2. In order to prevent this problem, the downstream condition must be satisfied.

$$cost(n_i, d) < cost(s, d) \tag{2.3}$$

An example is illustrated in Figure 2.7c where R2 is used as the downstream LFA for re-routing packets from R1 to R5.

- **Equal-Cost Multi-Paths Condition**—trivially, if Equal-Cost Multi-Paths (ECMP) exist, either of these paths can be employed for normal routing while the other can be used in the presence of failures without creating routing loops. Let $w(i, j)$ be the weight of link (i, j), the condition for ECMP is expressed as follows:

$$w(s, n_i) + cost(n_i, d) = w(s, n_p) + cost(n_p, d) \tag{2.4}$$

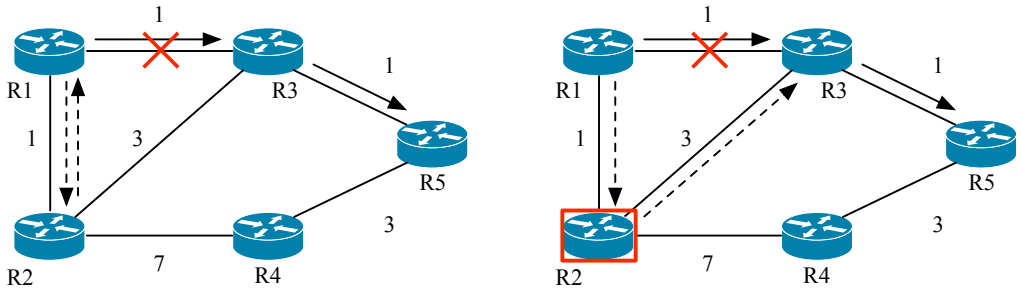
In practice, ECMP is often enabled by network operators to distribute the traffic load over multiple paths. Furthermore, if more than two equal-cost multi-paths exist, a router may store additional information on the next hops to allow multiple failures protection.

LFAs are desirable for real implementations for several reasons. First, it is easy to implement as there is no additional computation required apart from the condition validation. Second, a router employing LFAs does not require any excessive memory overhead as it only needs to enhance the existing routing table entry with additional information about the alternate next hops. That is, no additional routing entry is added. Third, no changes are required in the forwarding plane and packets can be re-routed immediately without being processed. Furthermore, even if some conditions may have a wider repair coverage than others, network operators can employ conditions that suit their purposes.

U-Turn Alternates

It is practically important for a network to elevate its reliability by employing resilient mechanisms to accommodate sensitive applications. LFAs alleviate packet loss rates by using pre-computed alternate next hops for fast re-route. Their repair coverage, albeit simple implementations, heavily depends on the topology. Thus, the reliability problem is not completely solved using LFAs.

Due to this problem, U-turn alternates [7] have been proposed to increase the repair coverage of LFAs, which are often low in practical topologies. Figure 2.8a illustrates an example of a scenario without an eligible LFA for re-routing packets to the destination. The notion behind U-turn alternates is that although the neighbour may not be used as an LFA, its adjacent nodes can potentially provide loop-free paths to the destination.



(a) No LFA available. (b) U-turn alternate.

Figure 2.8: An example of U-turn alternate.

Figure 2.8b shows that R1 can re-route packets to R5 via its eligible U-turn alternate, R2. In traditional routing paradigm, forwarding packets destined for R5 to R2 creates a loop as R2 is R1's child in the shortest path tree rooted at R5. However, R2 recognises that the traffic forwarded from R1 is a U-turn traffic using explicit or implicit packet marking or port detection.

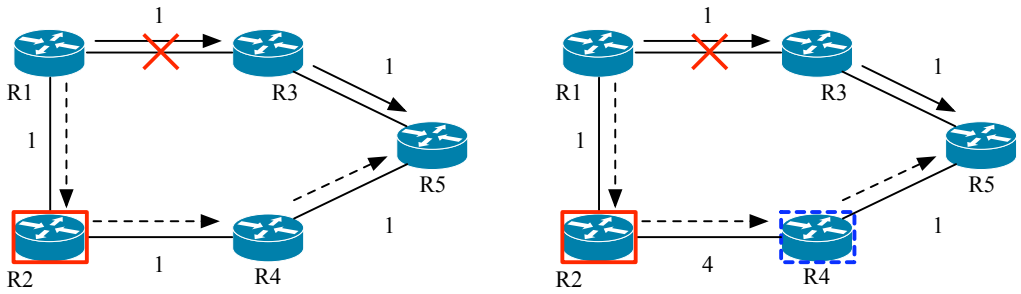
For example, R2 knows that R1 is its normal next hop to R5; hence, when R2 receives packets to R5 from R1, it forwards them to its available LFA, which is R3. In contrast, this local consideration can be omitted if R1 identifies the re-routed traffic by marking the packets. Additionally, a technique described in [131] can also be used to identify the U-turn traffic.

Tunnels

As its name implies, tunnels [17] employ the process which requires encapsulation and de-encapsulation. The failure-detecting node usually encapsulates the packet with a destination where packets are forwarded along the shortest path to the destination. However, it must ensure that after the packets are de-encapsulated at end points, they do not traverse back and create a forwarding loop.

In addition, directed forwarding can be used to avoid micro-loops caused by releasing packets at the point where routing through the shortest path tree involves the failure. Directed forwarding method uses two sets called *F-space* and *G-space* to identify a set of routers that are reachable by the detecting node without passing through the failure, and a set of routers that can reach the destination without traversing the failure. If there exists a node pair consists of one node from F-space and another node from G-space with a direct link, it can be used as tunnel-release end points [17]. Furthermore, it is important to note that, even if the directed forwarding is employed, some node pairs may not have alternate paths available.

An example of fast re-route using tunnels is shown in Figure 2.9. In Figure 2.9, when R1→R3 fails,



(a) Tunnel from R1 to R2, release at R2.

(b) Tunnel from R1 to R2, release at R4.

Figure 2.9: Examples of IPFRR using tunnels.

R1 encapsulates packets with R2's address[28]. Once R2 receives the packets, it performs de-encapsulation and forwards it via the optimal shortest path: R2→R4→R5. However, in Figure 2.9b, the weight of a link connecting R2 and R4 increases to 4. Therefore, routing traffic via the shortest paths when packets arrive at R2 can cause a micro-loop between R1 and R2. Under directed forwarding, R2 is specified as tunnel end point while R4 is its corresponding release end point. This allows packets to be forwarded along the path: R1→R2→R4→R5 correctly.

Not-Via Addresses

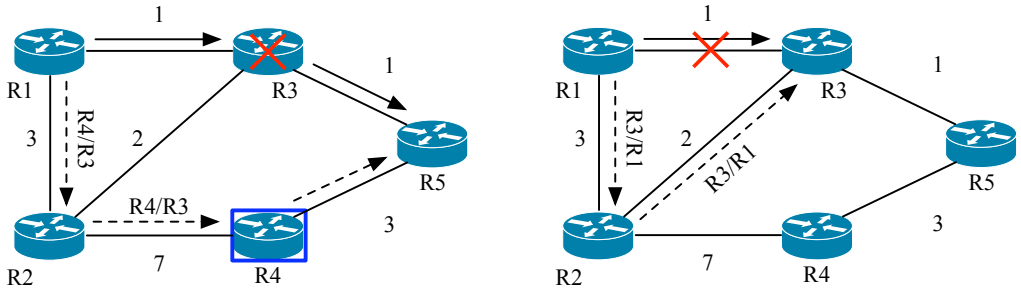
The usage of not-via addresses for fast re-route is defined in [18]. These addresses are typically used to protect specific interfaces in the presence of failures. Thus, for each interface in the network, two IP addresses are required (*i.e.* normal IP address and its corresponding not-via address). In general, not-via addresses are employed to protect packet losses by deviating the traffic around the failed elements. The packets encountering a failure must be encapsulated with not-via addresses[29]. These not-via addresses specify the failed nodes so that they do not contain in the repair paths.

However, if the failure is caused by the link connecting the detecting node and the destination, the computation of not-via address is performed by assuming a link failure instead of a node failure. This method ensures that the delivery is guaranteed for all recoverable failures. The main benefit of fast re-route using not-via addresses is a 100% repair coverage for any single failures, which can be either link or node failures. Nevertheless, most repair paths are pre-computed to avoid the node failures; hence, they are considerably longer than the optimal shortest paths after network re-convergence.

Figure 2.10a illustrates an example of using not-via addresses. When R3 fails, R1 encapsulates packets with a not-via address to R4 avoiding R3. Once R2 receives encapsulated packets, it selects the path to R4 that does not involve R3. After packets reach R4, they are de-encapsulated and forwarded to R3 using the normal shortest path. Alternatively, if the destination is R3 as shown in Figure 2.10b, the not-via address is computed in order to avoid R1, which allows packets to be re-routed to R3 successfully

[28] In practice, if the tunnel endpoints are adjacent to the detecting routers, encapsulation is not necessary.

[29] Note that, not-via addresses require IP-in-IP tunnelling, which may degrade the performance of routers.



(a) Re-route using not-via address from R1 to R5.

(b) Re-route using not-via address from R1 to R3.

Figure 2.10: Examples of IPFRR using not-via addresses.

if the root cause of the failure is the link.

Although in practice not-via addresses are complex compared to other IPFRR techniques, they can be implemented in conjunction with simpler methods such as LFAs to guarantee a full repair coverage while minimising the tunnelling operations. Recently, an investigation has been made on the combination of LFAs and not-via addresses [74]. It shows that combining these techniques does not result in any significant benefits. Thus, it is suggested that either technique is employed homogeneously.

In practice, employing not-via addresses does not only increase the computational and memory overheads, but also the complexity in the management plane due to re-routed traffic. Thus, a collection of techniques: not-via aggregation, prioritised not-via computation, and rNot-via have been proposed to improve the efficiency and manageability of the not-via addresses [66].

The first and second techniques allow the not-via addresses that are similar to their normal forwarding addresses to be aggregated; thus, the amount of memory required is reduced and the recovery time is shorten. It has been observed that the impacts on the network performance grow proportionally to the network size. That is, large networks benefit more from the not-via aggregation. Furthermore, rNot-via algorithm allows a router to determine whether it is part of the protection path of provided by not-via addresses so that the amount of re-route traffic can be estimated [66].

Failure Insensitive Routing

Most IPFRR techniques are based on the traditional routing paradigm where a router forwards packets according to the IP addresses indicated in the header. Failure Insensitive Routing (FIR) [93], however, offers a 100% repair coverage for any recoverable single link failures by employing an interface-specific forwarding table. It suggests that the notifications generally propagated once a failure is detected should be suppressed for a certain duration to ensure that it is not intermittent. The notion of FIR is most failures are transient [52, 73] and it is redundant to trigger the convergence process realising that the failure can be potentially recovered before the process completes.

Under FIR, routing is performed normally in the failure-free case. In the presence of failures, the detecting router suppresses the failure notification and forwards the packets to affected destinations

using the backwarding table. It has been proved that without multiple failures, FIR can re-route packets successfully without jeopardising other network operations. Furthermore, it has been observed that FIR reduces the routing instabilities due to intermittent failures and provides higher availability in networks that employ link-state routing protocols such as OSPF and IS-IS [93].

SafeGuard

In general, fast re-route in IP networks can be achieved through pre-computed backup paths. These backup paths are typically computed in regard to local failures. However, a routing technique known as SafeGuard [67] computes the backup path based on a different approach.

Under SafeGuard, packets are forwarded normally under a failure-free case. In the presence of failures, SafeGuard employs Cost-Carrying Packets (CCP), which contain the information about the remaining cost of the path towards the destination. Basically, the detecting node embeds the cost of the backup path into the packet header and forwards it to the alternate next hop. When each intermediate router receives a CCP, it determines whether the cost is consistent with the local cost to the destination. If the result is negative, it selects the path that matches with the cost and replaces it with the remaining cost. Thus, CCP can be re-routed to the destination successfully without traversing through the failed element.

Although routers employing SafeGuard are subject to an increase in packet processing overhead, it has been proved to be negligible. Furthermore, when equal-cost paths exist, the computation of backup paths become more complex as the costs of links must be altered to prevent cost collision [67]. SafeGuard also requires considerable amount of memory overhead to store all possible costs per destination.

2.3.4 Multi-Topology and Multi-Path Routing

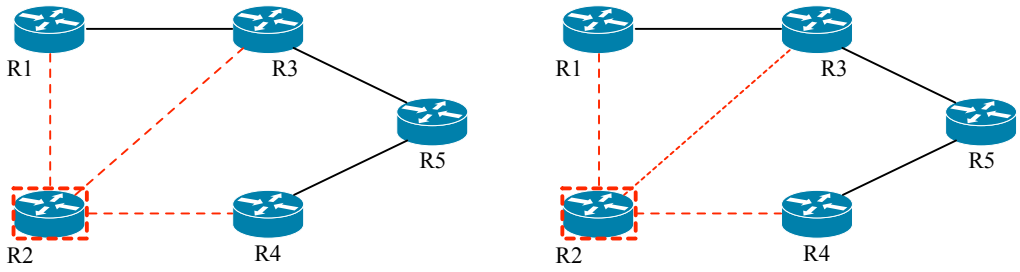
Traditionally, a network needs to re-converge when there is a change in topological information to retain a consistent routing. That is, all routers share the same network map. Multi-Topology (MT) and multi-path routing allow packets to be delivered via different network maps/paths and hence, increase the reliability. The followings present different existing strategies for MT and multi-path routing.

Multiple Routing Configurations

Multiple Routing Configurations (MRC) [63] is based on enhancing the routers with additional information based on different configurations. The technique falls into both MT routing and IPFRR as it also provides fast repair paths. The main advantages of MRC are loop-free environment and a full repair coverage for any recoverable single failures without knowing their root cause. Furthermore, it can be implemented without any major modifications to existing standards.

The computations of the MRC are performed in advance to ensure fast recovery. However, these computations can be done offline using mechanisms proposed for MT routing [100, 99]. MRC calculations are based on isolating nodes and links from at least one configuration without partitioning the network to ensure that MRC covers all single failure scenarios. While there is an issue concerning the scalability of MRC, the typical number of configurations required is 3–4 in average for practical topologies [63]. Furthermore, MRC offers better load distribution in post-failure scenarios; hence, it is

considered to be practical [61].



(a) Isolating R2. (b) Isolating R2–R3.

Figure 2.11: Examples of configurations in MRC.

Figure 2.11a shows that the node R2 is isolated; thus, all links connecting R2 to other nodes becomes restricted. That is, only traffic originated and sent to R2 can employ these links. In contrast, Figure 2.11b illustrates an isolated link R2–R3. Thus, no traffic including transit traffic can traverse through R2–R3.

Recently, an IPFRR scheme called relaxed MRC (rMRC) has been proposed to simplify the process of computing the configurations in conventional MRC [21]. The technique also increases the routing flexibility. In rMRC, not all links need to be isolated as it employs the adjusted forwarding procedure, which minimises the number of configurations. The performance of rMRC is better than MRC in every aspect including overheads (*e.g.* number of configurations) and impacts on network traffic (*e.g.* MLU).

Resilient Routing Layers

A technique known as Resilient Routing Layers (RRL) [62, 46] has been proposed to restore the packets from failures. The strategy used to provide protection against failures is similar to that of MRC. In RRL, each node has a subgraph of a network topology known as *safe layer*. This layer is used to handle traffic in the presence of failures.

In contrast to MRC where backup configurations are created by assigning different link weights to the elements in the network, RRL completely removes the link from the original topology to create a backup topology [38]. Similarly, re-routed packets under either MRC or RRL must be marked with the topology identification to maintain a consistent routing.

Furthermore, the distribution of topology information and forwarding process for both RRL and MRC are conformed to the mechanisms specified in MT-OSPF for IPv4 [100] and MT-OSPFv3 for IPv6 [84]. In contrast, M-ISIS [99] recommends that the Differentiated Services Code Point (DSCP) field is used for topology identification.

Beyond MRC and RRL, other works based on MT routing for network resilience have been proposed. For example, it is possible to employ different topologies to deviate the packets from the path that incorporates the failure [81]. However, the proposal does not specify any new strategy for generating

topologies. In addition, an investigation has been made on the optimal number of different topologies required to provide full protection for any recoverable link failures [107].

Routing Deflections

The approach of routing deflections is very similar to multi-path routing [131]. Intuitively, packets are deflected from the optimal shortest path when a failure occurs. However, it has to retain the loop-free property to guarantee successful delivery and avoid unnecessary bandwidth consumption. In order to employ routing deflections, there must be a set of constraints used as criteria for selecting neighbours that can be used for packet forwarding. Moreover, there must be an explicit signal to inform routers of the chosen path for each packet. Deflection rules are defined in [131] as follows:

- **One hop down**—neighbours that have path costs to the destination less than the path cost from the deflecting node to the destination can be used as loop-free next hops.
- **Two hops down**—neighbours that are one hop down (*i.e.* downhill) or have path costs to the destination less than the path cost from the previous hop to the destination (*i.e.* two hops down) can be used for routing deflections.
- **Two hops forward**—neighbours that are not previous hops and satisfy either downhill or two hops down condition can be used for routing deflections.

Furthermore, in order to enable routing deflections, a 6-bit tag information must be stored in the IP Identifier field. A randomised modulo arithmetic method is used to map these tags to deflections. This ensures the degree of freedom of the deflections [131]. Since routing deflections provide significant path diversity for all node pairs, the reliability of the network is increased accordingly.

Multi-Path Inter-Domain Routing

Most routing strategies mentioned previously are used to handle failures in a single domain. In contrast, a technique called Multi-Path Inter-Domain Routing (MIRO) [129] has been proposed to reduce the losses in the inter-domain level based on the multi-path approach.

Basically, routing in the inter-domain level relies on the BGP, which restricts the domain from employing multi-paths for each destination prefix. Under MIRO, routers learn the normal BGP routes. In addition, it allows ASes to negotiate and employ additional paths to satisfy their requirements such as reliability and security [129].

Although BGP has extensions to enable multi-path advertisements for the same destination prefix, the route selection process has not been described in details [127]. Thus, it is possible for MIRO to employ these extensions for route advertisements.

Path Splicing

Path splicing is a technique used to provide resilient routing in a network through multiple routing trees known as *slices* [88]. In path splicing, different alternate paths (*i.e.* slices) are generated to offer path diversity. Each packet carries the splicing bit which is used to determine the correct routing tree. Furthermore, packets can switch from one slice to another at intermediate hops, which can be directly

controlled by end systems. Thus, if a failure occurs, the failure-detecting end system can re-route the traffic via other slices that are still operable.

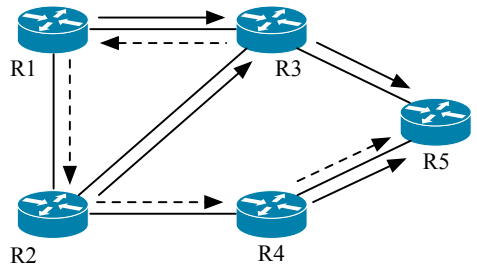


Figure 2.12: An example of path splicing with two slices.

Figure 2.12 illustrates an example of a network with two slices rooted at R5 (*i.e.* two routing trees). In path splicing, it is possible for R1 to forward packets to R5 via R2 and R3 which belong to different slices. These slices are computed using the degree-based perturbations of link weights. That is, the weight of each link is modified according to the degrees of nodes at both ends. However, the method for computing alternate paths are not restricted; thus, other algorithms may be used.

A comparison with routing deflections made in [88] shows that path splicing offers lower stretch alternate paths. Unlike routing deflections where the number of neighbours can be significantly high, the memory overhead of path splicing is bounded by the number of slices. In addition, employing path splicing in practical networks such as Abilene and Sprintlink has minimal impacts on the network traffic. Although infrequent routing loops are possible under path splicing, persistent loops can be avoided using network-based recovery [88].

2.3.5 Overlay Networks

A specific type of overlay known as the Resilient Overlay Network (RON) [4] has been developed to allow applications to detect and recover packets from failures, which disrupt the forwarding continuity. It enhances the end hosts with an ability to exploit the forwarding paths, which are not available in traditional routing. RON nodes are required to perform continual probing of alternate paths; hence, they do not offer a fast recovery as in IPFRR. However, the repair paths provided by RON can significantly reduce the time to detect and react to inter-domain failures in order of seconds. For example, if a direct connection between two RON nodes exists, after the probing, RON nodes can send the traffic to each other directly as an alternate to the normal path. This concept also applies to the failure scenarios where one or more paths are unavailable.

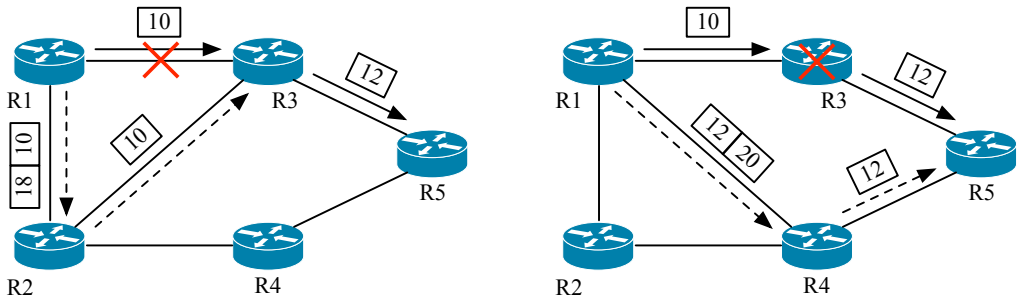
A case has been made on RON to ensure its practicality [3]. It has been observed that RON can increase the end-to-end reliability in the Internet through the employment of alternate paths. Although initially, RON is subject to scalability problem, a technique called Destination-Guided Detouring via Resilient Overlay Network (DG-RON) [102] can be applied to alleviate the problem by simplifying the

path exploration based on detour sets. Thus, the frequency of alternate paths probing under RON can be significantly reduced without sacrificing the path diversity. However, RON is still subject to different issues regarding instabilities, expression of routing policies, route selection, and interaction between RON nodes [3].

2.3.6 MPLS-Based Resilience

In general, MPLS can enhance the reliability of a network by enabling the RSVP Traffic Engineering (RSVP-TE), which provides MPLS local restoration via MPLS Fast Re-Route (MPLS-FRR) [97]. The technique is commonly used to recover packets from link or node failures by employing backup paths. It allows network planners to establish different paths across backbone routers that comply with the routing policies [82].

Concisely, MPLS-FRR for link protection requires a router to set up the next hop backup tunnel. The failure-detecting node known as the Point of Local Repair (PLR) is responsible for swapping the normal label and pushing the backup labels in order to deviate the packets from the failed link. Generally, the backup path terminates at the Merge Point (MP) where the traffic can be forwarded along the primary path. For node protecting MPLS-FRR, the next-next hop tunnel is required. At PLR, the next hop label needs to be swapped while the backup label is pushed into the packet header. Similar to link protection, the traffic can be forwarded through the primary path once the backup path overlaps the normal path [97].



(a) Link protection.

(b) Node protection.

Figure 2.13: Examples of MPLS-FRR.

Figure 2.13 illustrates link and node protection examples of MPLS-FRR. Furthermore, MPLS can be used to enhance the reliability of OSPF networks by employing virtual links through tunnels (*i.e.* MPLS paths). These links are treated as physical links and are taken into account when performing path computation. It is important that tunnels are set up to increase the degree of freedom of ECMP, which can be used in the presence of failures. The technique is commonly known as Protection using OSPF-ECMP with MPLS (POEM) [53].

2.3.7 Disjoint Paths and Redundant Trees

Employing either link or node disjoint paths can increase the reliability of the network as packets can be forwarded along different paths that are maximally exclusive (*i.e.* if one fails, another is still operable). A large number of algorithms has been proposed to compute disjoint paths [120, 121, 12, 111, 124, 95, 114].

Network failures can be recovered using *k-shortest disjoint paths* [30]. Even if the computational overhead of disjoint paths computation can be significant in complex networks, it can be optimised using the probabilistic methods which enable distributed computation [15]. Furthermore, managing and optimising the disjoint paths are difficult [45]; thus, it is recommended that a network is divided into subtopologies to permit structured recovery mechanisms [55, 56]. Regardless of its simplicity, failure recovery using disjoint paths requires a proper protocol that allows routers to communicate their employing paths [56].

Most redundant trees are generally, structured recovery schemes of k-disjoint paths. Basically, these techniques construct different trees to create path redundancy. These trees are commonly disjoint known as *red* and *blue* trees. Many algorithms are used to construct redundant trees in networks that are specifically strongly connected [77, 76, 130]. In addition, although redundant trees were originally introduced to increase the reliability in optical networks, it has been investigated that the concept is also applicable with MPLS networks [9].

2.3.8 Protection Cycles and Pre-Configured Cycles

Protection cycles and pre-configured cycles (p-cycles) have been proposed to elevate the reliability in ring and mesh-based networks. Protection cycles are capable of restoration of link failures by switching cycles for traffic forwarding [31]. In addition, an efficient algorithm known as Hamiltonian cycle protection has been proposed to reduce the complexity of the protection cycles by aggregating all cycles into a single spanning ring [49].

Under p-cycles, a network is divided into different cycles. For each cycle, a single failure can be recovered using the direction opposite to the normal path [43]. In certain scenarios, p-cycles can restore packets from dual link failures [42]. Although p-cycles were originally introduced to provide protection in optical networks, it can be used to guarantee the bandwidth in MPLS networks [57]. In addition, the technique can be applied to IP-based networks using virtual circuit techniques to form closed logical loops known as virtual protection cycles [117]. However, p-cycles restoration paths are very inefficient due to their significant path length stretch, which is unacceptable in IP networks [62].

2.3.9 Eliminating the Convergence Process

Most approaches focus either on slowing down the convergence process and re-routing the traffic during transient period or speeding up the convergence process and preventing transient loops to minimise packet losses. However, another approach has been proposed to completely eliminate the convergence process by employing Failure-Carrying Packets (FCP) [65].

Under the routing technique employing FCP, packets are forwarded normally in the failure-free case. When the packet encounters a failure, the router embeds the failure information, typically the link

label, into the packet header. The failure-detecting router is also responsible for calculating the new path for the FCP. All intermediate nodes of the new path are required to perform similar process to ensure that packets are forwarded correctly without traversing the known failures. If the FCP encounters another failure, additional information on the failed link is embedded into the packet header. Although methods for optimising the packet and computational overheads have been proposed, the space required to permit labels embedding is still significant. Employing FCP has a huge benefit regarding the reliability of a network since embedding failure information into the packet header allows routers to deviate packets from all known failures.

2.4 Conclusions

Routing is one of the most important operations in data communications. The process is used to select paths for the traffic between any source-destination pairs. In general, when a packet arrives, the router performs the table lookup to decide its next hop. This routing table can be constructed differently based on several algorithms such as Dijkstra's algorithm and Bellman-Ford algorithm.

Basically, the Internet routing is divided into intra-domain and inter-domain levels, as a single routing protocol cannot accommodate all operations. For example, OSPF and IS-IS are link-state routing protocols employed to perform routing within a single AS based on the shortest paths. However, as each router requires the information on the entire network map, the protocols cannot serve as inter-domain routing protocols due to scalability problem. Thus, routing protocols such as BGP, which is based on path vector paradigm is used to provide features that satisfy the requirements of ISPs. Under BGP, each domain can apply different routing policies that are unknown to the others to accommodate several issues such as business models and security.

As new technologies emerge (*e.g.* IPv6), routing protocols need to provide compatibility. Several extensions have been made to existing protocols to minimise the demand for changes in the current infrastructure. Furthermore, even if some protocols provide better performance and more features, it is greatly believed that other routing protocols will remain in the Internet for sometime due to their simple implementations. Ultimately, the decision on protocols selection lies on network operators.

This chapter pointed out that the mechanisms employed in the current networks are not sufficient to accommodate emerging services, mainly due to the conventional convergence process. Thus, topological changes such as link or node failures can lead to a vast amount of packet loss. Several approaches have been proposed to elevate this problem. Tweaking protocol parameters to urge the convergence process results in a faster failure detection, but it may incur routing instability in case of intermittent failures. In addition, a loop-free convergence mechanism such as oFIB reduces the number of transient loops during network re-convergence. However, these techniques do not completely eliminate the packet losses problem.

IPFRR techniques such as LFAs, U-turn, tunnels, not-via addresses, and FIR employ pre-computed alternate paths for fast re-route in the presence of single failures. Similarly, MRC and RRL use the backup topologies for failure recovery. Path splicing is a different resilient approach where a network is divided into slices where routes can be changed at intermediate nodes. These mechanisms can signifi-

cantly elevate the network reliability.

Furthermore, deflecting routes from the optimal shortest path can increase the path diversity, which is important for network resilience. Most techniques presented aim to enhance the performance of a single domain. However, MIRO extends the BGP's capability to employ multi-path between negotiating ASes to guarantee a reliable connection between networks.

In practice, not all nodes are connected through ISPs and may have direct connections. Thus, RON uses probing to detect available paths which are employed as backups when there is a failure. Routing resilience in MPLS networks can achieve fast re-route feature as in IP networks by enabling its RSVP-TE feature. Alternatively, developed techniques based on disjoint paths and redundant trees are available for deployment in MPLS networks. For optical networks, protection cycles and p-cycles are valid resilient mechanisms but are considered unacceptable in IP networks due to their low efficiency (*i.e.* very high stretch alternate paths).

Recently, FCP has been proposed as an alternative approach where packets carry information about the failures they have encountered. The routers use this information to compute appropriate paths that do not involve known failures; hence, packets can be delivered successfully even though there are multiple failures in the network.

Chapter 3

Enhanced Loop-Free Alternates

3.1 Introduction

Throughout this work, it is presented that network reliability problems involve the minimisation of packet loss in the presence of failures. The emerging services cannot afford to rely on traditional routing paradigm due to unavoidable damage caused during re-convergence. This increases the demand for a highly reliable network.

Several approaches such as multi-path and multi-homing routing [88, 129, 131] and overlay networks [4] have been proposed to alleviate this problem. In this chapter, the fast re-route and recovery approach defined in the IP Fast Re-Route (IPFRR) framework [113] is focused. It specifies two main components for providing a disruption-free forwarding which are fast failure detection and fast re-route mechanisms. Achieving fast failure detection can be done by tweaking the protocol parameters [39] (*i.e.* setting an appropriate value for HelloInterval). Nevertheless, the amount of packets being dropped from the time an actual failure occurs until it is detected by a router is unavoidable. Thus, it is important to employ a mechanism that permits a router to re-route the traffic for affected destinations via other paths. Several analyses [52, 73] show that most failures are transient (*i.e.* short-lived) and more than 50% last less than a minute. Consequently, most IPFRR techniques [93, 8, 18] aim to handle transient failures.

Recently, many techniques such as Loop-Free Alternates (LFAs) [8], U-turn [7], tunnel [17], not-via addresses [18], and Failure Insensitive Routing (FIR) [93] have been introduced. It is greatly believed that LFAs are the most feasible solutions for resilient routing. Nevertheless, their repair coverage depends heavily on the underlying network, which can be as low as 60–70%.

This chapter presents the Enhanced Loop-Free Alternates (E-LFAs) which are based on the normal LFAs. The approach employs a simple recursive method on the normal LFAs to enhance their repair coverage. E-LFAs allow a router to re-route packets through a number of alternate next hops until it can be forwarded along the shortest path to the destination, given that no immediate LFA is available. The followings summarise the main features of a routing strategy using E-LFAs:

- It provides a near optimal repair coverage for an arbitrary network.
- It does not incur any either heavy computational or memory overhead.
- It re-routes packets from single failures and preserves the loop-free environment.

- It does not require any major modifications, which makes it easy to implement.
- It does not jeopardise the performance of a router.

3.2 Computing Enhanced Loop-Free Alternates

E-LFAs employ similar conditions used in LFAs. If an eligible alternate next hop candidate exists for any node pair, it is used to re-route packets normally according to the basic specification of LFAs [8]. Nevertheless, this is not always the case. For most practical topologies, it is often that none of the neighbours of the failure-detecting node can be used to re-route packets to the destination without creating a forwarding loop. E-LFAs extend the repair coverage of normal LFAs by using a simple recursive method.

Algorithm 3.1 Computing E-LFAs using a simple recursive method.

Input: s, d, N, L(d)

Output: $E(d)$

1: $E(d) =$ **null**
2: **for all** $n_i \in N$ **do**
3: $C = \emptyset$
4: $count \leftarrow 0$
5: $current\_node \leftarrow n_i$
6: **while** $current\_node \notin C$ **do**
7: $count \leftarrow count + 1$
8: $C \leftarrow C \cap current\_node$
9: **if** $LFC(current\_node)||NPC(current\_node) ==$ **true then**
10: $E(d) \leftarrow E(d) \cap (n_i, count)$
11: **break**
12: **else**
13: $current\_node \leftarrow LFA(current\_node, d)$
14: **end if**
15: **end while**
16: **end for**
17: **return** $E(d)$

As described previously in section 2.3.3, the neighbour nodes can be classified by their abilities as alternate next hops. However, E-LFAs concern only two types of LFAs which are: *a)* Loop-Free Condition (LFC), and *b)* Node-Protection Condition (NPC).

The notion of having other conditions such as Downstream Condition (DSC) is that network operators may want to avoid short period of forwarding loops in the presence of multiple failures. Nevertheless, E-LFAs do not incur such problem which lavishes the network capacity.

Let $G = (V, E)$ be the graph with vertices $V = \{v_1, v_2, ...\}$ and edges $E \in V \times V$ representing the network topology. Normal LFAs of all sources to destination d are denoted as $L_0(d) = \{LFA_0(v_1, d), LFA_0(v_2, d), ...\}$. To compute E-LFAs, $L(d) = \{LFA(v_1, d), LFA(v_2, d), ...\}$ represents a set of LFAs from all source nodes to d where $LFA(v_i, d)$ is equal to $LFA_0(v_i, d)$ or the primary next hop of v_i, $n_p(v_i, d)$, if no eligible LFA exists.

Given a source s with a set of neighbours without $n_p(s, d)$, N, and a destination d, E-LFAs can be computed using Algorithm 3.1.

Consider a failure-detecting node s. If a destination d has no normal LFA, s considers each neighbour in N to find a set of eligible E-LFAs with their corresponding counters, $E(d)$. Each neighbour node, n_i, is considered. If it does not satisfy the condition (LFC or NPC), $LFA(n_i, d)$ becomes a candidate. If this does not meet the condition, the same process is repeated. The iteration continues until either a node along the path via a number of alternate next hops satisfies the condition or a loop occurs in the path (*i.e.* the path traverses the same node more than once). For the latter case, n_i cannot be used as an E-LFA. If n_i is an eligible E-LFA, it is added into $E(d)$ with its corresponding $counter$. This counter determines the number of repetitions a packet must traverse through a series of alternate next hops of intermediate nodes until it can be forwarded along the shortest path again. Once the algorithm completes, s determines all nodes in $E(d)$ and selects the node that provides the shortest path to d as E-LFA.

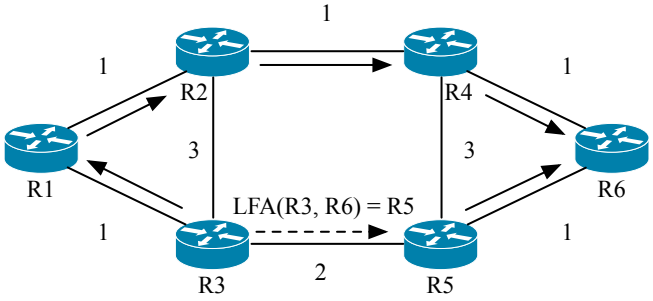


Figure 3.1: A simple network topology illustrating LFA from R3 to R6.

A simple network topology used to illustrate E-LFA computation is shown in Figure 3.1. It can be seen that all nodes in the network except R1 have eligible LFAs to R6. For example, if link R3→R1 fails, R3 can forwards the packets destined for R6 via R5 successfully without creating any forwarding loop. However, if link R1→R2 fails, all packets traversing via R1 to R6 are dropped until the network re-converges. This potentially incurs a significant amount of packet loss. With E-LFAs, R1 determines its neighbour node, R3, which is not its primary next hop to R6. Since LFA from R3 to R6 (*i.e.* R5) satisfies the LFA condition, R1 employs R3 as its E-LFA to R6.

It is important to note that, U-turn [7] has similar approach where the alternate next hops of neighbour nodes are taken into consideration. However, U-turn makes use of the immediate neighbours only

while E-LFAs are capable of creating alternate paths via both nodes with and without eligible LFAs. This results in E-LFAs having a higher repair coverage. Furthermore, since network failures are not limited to a single type, it is often worth enhancing the routing table entries with information for each type of failures. However, the ultimate decision belongs to the network operators.

3.3 Packet Processing and Forwarding

In the normal case, a router forwards the packets via the shortest path. When a failure occurs, the detecting node forwards the packets to affected destinations using LFAs. However, for destinations without normal LFAs but E-LFAs, the detecting node marks each packet with the number of repetitions it has to be forwarded along the alternate next hops. When an intermediate node receives a re-routed packet, it decrements the number of repetitions by 1 and forwards it using its local LFA based on the destination. Once this number reaches 0, the packet can be forwarded using the normal shortest path. Algorithm 3.2 summarise the packet processing process.

Algorithm 3.2 Packet processing at node s.

Input: `in_pkt`

Output: `out_pkt`

1: **if** `in_pkt`.$counter == 0$ **then**
2: **if** $(s, n_p(s, \texttt{in\_pkt}.d)) == \text{failed}$ **then**
3: `in_pkt`.$counter \leftarrow counter(s, \texttt{in\_pkt}.d) - 1$
4: **return** `out_pkt` $\leftarrow$ `in_pkt`
5: **else**
6: **return** `out_pkt` $\leftarrow$ `in_pkt`
7: **end if**
8: **else**
9: **if** $(s, n_s(s, \texttt{in\_pkt}.d)) == \text{failed}$ **then**
10: Drop(`in_pkt`)
11: **return** null
12: **else**
13: `in_pkt`.$counter \leftarrow$ `in_pkt`.$counter - 1$
14: **return** `out_pkt` $\leftarrow$ `in_pkt`
15: **end if**
16: **end if**

E-LFAs omit the consideration of DSC and ECA conditions without creating any pitfalls. The routing technique employs an additional bit to indicate a re-routed packet. Therefore, if a re-routed packet encounters another failure, it is dropped immediately to avoid transient loops[1].

[1] If multiple failures occur, a re-routed without indication may be caught in a loop until its TTL expires. This results in a network being utilised futilely.

3.4 Termination of Using LFAs and E-LFAs

When a failure occurs, the detecting router is responsible for advertising the failure to the rest of the network. A router receiving this information updates its database and determines whether any paths to destinations are affected by the failure. If one or more destinations are affected, re-calculation of shortest paths is mandatory. The means for employing LFAs and E-LFAs are similar as they allow packets to be re-routed immediately without waiting for the completion of network re-convergence. However, a router using LFAs basically limits the amount of time of the alternate next hops usage. After that, it continues to use new primary next hops based on the new network map [8]. This avoids possible transient loops during convergence process.

On the other hand, the termination of using E-LFAs cannot be determined by confining the amount of time. If the network converges, new LFAs and E-LFAs are re-computed. Therefore, re-routed packets that have been marked but not delivered may be forwarded incorrectly. Routing via E-LFAs is performed with an assumption of intermittent failures. Therefore, it suppresses the failure notification for a period of time. If the failure persists, a router must trigger the convergence process. In addition, several techniques are recommended [134, 14] for minimising possible micro-loops during network convergence.

3.5 Properties

The key property of E-LFAs is loop-free forwarding. Trivially, routing under failure-free case is loop-free as E-LFAs employ normal shortest path routing where all routers share a consistent network topology. For recoverable failure scenarios, routing can be divided into two cases: *a)* one or more LFAs exist and *b)* one or more E-LFAs exist. Theorem 3.1 and Theorem 3.2 describe the loop-free characteristic of routing under LFAs and E-LFAs.

Theorem 3.1. *Given a destination d, routing in the presence of a single failure via LFA of the failure-detecting node, s, does not cause the packet to traverse any link more than once.*

Proof. Although there are many types of LFAs specified in the basic specification [8], only LFC and NPC are determined. Equation 2.1–2.2 are criteria for link and node protection LFAs.

Let s be the failure-detecting node and d be the destination. If the path from a neighbour of s, n_i, to d is shorter than the path from n_i to d via s, a path from n_i to d does not involve s. Thus, the path does not involve the failed link $s \rightarrow n_p(s,d)$ where $n_p(s,d)$ is the next hop of the shortest path from s to d. Since packets are forwarded along the shortest path from n_i to d without traversing through the failed link, employing n_i to protect link failure is loop-free.

Similarly, if a path n_i to d is shorter than the path from n_i to d via $n_p(s,d)$, a path from n_i to d does not involve $n_p(s,d)$. This implies that the shortest path from n_i to d does not contain s. Therefore, employing n_i to protect node failure is also loop-free. □

Theorem 3.2. *Given a destination d, routing in the presence of a single failure via E-LFA of the failure-detecting node, s, does not cause the packet to traverse any link more than once.*

Proof. E-LFAs inherit the property of normal LFAs which provides a loop-free environment. The property can be proved by induction of the computation illustrated in Algorithm 3.1. In the process of finding E-LFA, if the node is previously considered by the algorithm, the neighbour of s, n_i cannot be used as an E-LFA and is discarded. Thus, if Algorithm 3.1 yields an eligible E-LFA, it implicitly guarantees a loop-free path. □

3.6 Performance Evaluation

This section presents the evaluation of E-LFAs. The broad areas of investigation include protocol overheads, characteristics of the alternate paths, the impact of failures and associated recovery mechanisms on link load.

Regarding the overheads, computational and memory overheads imposed on each router in order to compute and store LFA or E-LFA for each destination are considered. Furthermore, the number of bits required in the packet header for employing the forwarding mechanism is evaluated. To analyse backup path characteristics, their stretch and number of hops are considered and compared to *OSPF re-route*[2] which is used as a benchmark throughout this evaluation. The repair coverage, which is the main contribution of E-LFAs is compared with LFAs. It is important to note that, fast re-route mechanisms achieve the performance level reported immediately after a failure is detected, whereas OSPF re-route normally requires several seconds of re-convergence stabilisation before it can achieve its reported performance. Since neither LFAs nor E-LFAs can guarantee a full repair coverage, the post-failure traffic characteristics of the network is not considered.

3.6.1 Method

Self-implemented Java software model is created to compute LFAs and E-LFAs. The simulations are run on a machine with a 2.16 GHz Intel Core 2 Duo processor and 2 GB memory. This Java software are verified through both dynamic (experimentation) and static (analysis) verifications[3].

Table 3.1: Properties of topologies used in simulations.

Topology	Type	# of nodes	# of links	In-/Out degree
Abilene	Real	11	28	1.273
GEANT	Real	23	74	1.609
Abovenet (AS 6461)	Inferred	138	744	2.906
Sprint (AS 1239)	Inferred	315	1944	3.086
Tiscali (AS 3257)	Inferred	161	656	2.037
Waxman	Random	100	400	2.000
Barabasi-Albert	Random	100	394	1.970
Barabasi-Albert-2	Random	100	766	3.830

[2]The path used upon completion of the re-convergence process.

[3]All classes have been tested through functional test and the code are verified based on the code conventions.

The Abilene [133] and GEANT [37] topologies are used in order to obtain realistic results. To illustrate that E-LFAs can perform better than LFAs for an arbitrary network topology, Rocketfuel data [115] and synthetic topologies generated by BRITE [79] based on Waxman, Barabasi-Albert (BA) and Barabasi-Albert-2 (BA-2) models are also used.

As some inferred topologies have notable high path diversity (*e.g.* Sprintlink) [122] due to false links result from Domain Name System (DNS) misnaming [132], Abovenet, Sprintlink, and Tiscali are used instead of a single topology to ensure accurate results. In addition, the standard error is provided to ensure that the results are reproducible.

3.6.2 Overheads

The followings evaluate different types of overheads:

Computational Overhead

The computational overhead of E-LFAs are determined by the time complexity of computing E-LFAs and their corresponding counters. The results for different topologies are shown in Table 3.2. The complexity of Algorithm 3.1 is bounded by $O(V^2 \times E)$ for all nodes in the network, where V and E represent the vertices and the edges in the network. However, it can be seen that, for the largest network topology (*i.e.* Sprintlink), the computation time is lower than 2 ms.

Table 3.2: Repair coverage of different topologies under LFAs and E-LFAs.

Topology	**Link protection (%)**		**Node protection (%)**		**Time (ms)**			
	LFAs	**E-LFAs**	**LFAs**	**E-LFAs**	**LFC**	**StdErr**	**NPC**	**StdErr**
Abilene	65.455	89.091	58.537	85.366	0.029	0.001	0.039	0.001
GEANT	91.107	99.209	81.250	88.889	0.067	0.002	0.096	0.002
Abovenet	97.549	99.558	81.884	94.823	0.332	0.002	0.402	0.003
Sprintlink	96.242	98.815	77.733	95.417	1.383	0.004	1.516	0.003
Tiscali	88.063	97.205	78.792	93.391	0.402	0.003	0.461	0.003
Waxman	91.404	99.677	87.432	97.989	0.183	0.002	0.200	0.004
Barabasi-Albert	92.707	99.869	81.201	95.361	0.183	0.003	0.263	0.003
Barabasi-Albert-2	98.802	99.969	95.917	99.886	0.185	0.003	0.271	0.003

Memory Overhead

Similar to LFAs, E-LFAs do not require any additional routing table entries to provide fast re-route. However, the entry of each existing destination must be enhanced with information about the E-LFA and its corresponding counter. This does not incur any significant memory overhead and is generally considered a good trade-off for routing resilience.

Packet Overhead

Packet overhead can be evaluated by the number of bits required in the packet header in order to permit packet forwarding via E-LFAs. From simulation results, the maximum value of a counter required is 3.

However, more than 99.586% of E-LFAs require only 1 bit and at most 3 bits for link protection, and more than 99.943% of E-LFAs require only 2 bits and at most 3 bits for node protection. In addition, an extra bit is recommended for re-routed packet indication. Thus, re-routing via E-LFAs requires 4 bits to protect either single link or node failure cases. Since there are two versions of IP packets deployed in the current networks, a part of Type of Service (TOS) field in IPv4 packet and a part of Traffic Class field in IPv6 packet can be used.

3.6.3 Repair Coverage

The repair coverage in term of percentage of protectable destinations are illustrated in Table 3.2. It can be seen that E-LFAs improve the repair coverage of normal LFAs in all topologies regardless of the protection condition. In average, 7.758% more links and 13.547% more nodes are protected. However, it is important to note that, a router still employs alternate paths via LFAs if they exist. Performing the recursive method repeatedly can further increase the repair coverage in certain cases. However, the difference in performance is negligible compared to the additional memory and complexity required in the forwarding plane.

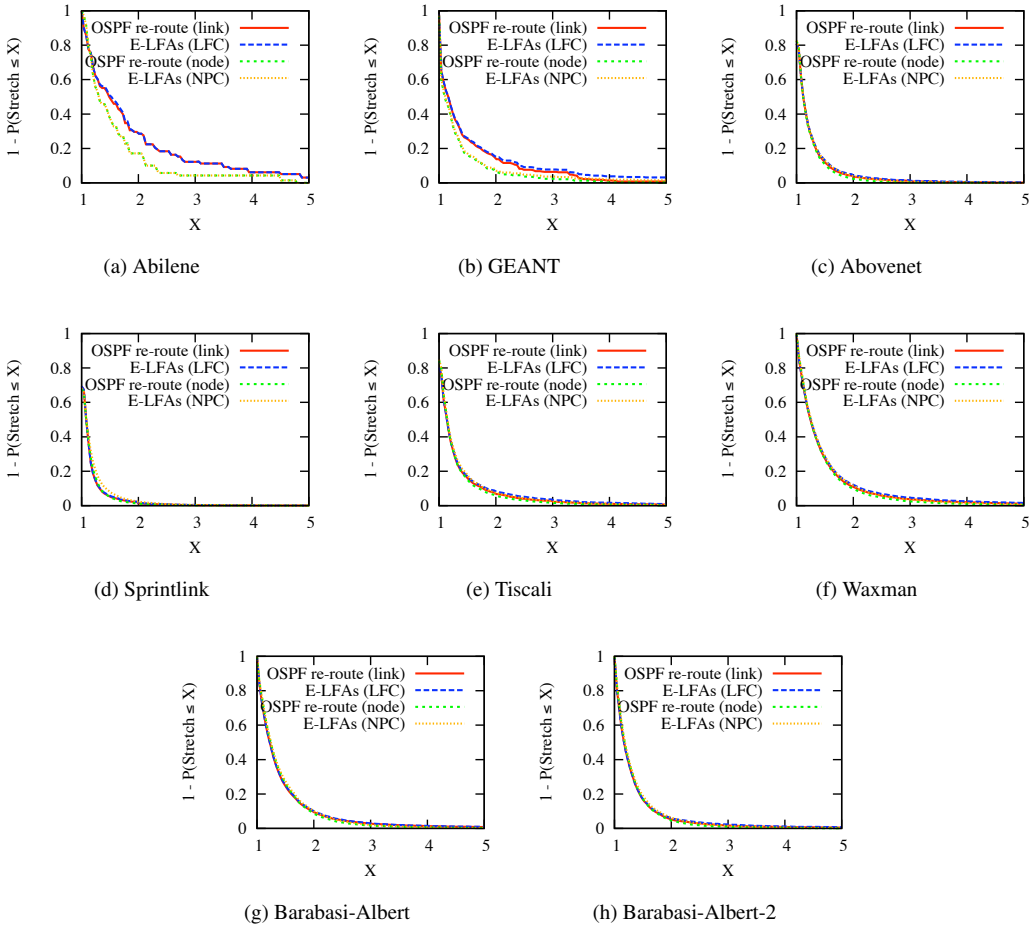
3.6.4 Stretch

Providing fast re-route via loop-free alternate next hops can alleviate forwarding discontinuation problem. In a network that employs only LFAs, the stretch of an alternate path is not significant as the path cost is basically the sum of the link cost connected the detecting node and its LFA, and the shortest path from its LFA to the destination. However, enhancing the repair coverage with E-LFAs may incur higher path length stretch as the packet needs to traverse via a number of alternate next hops before it can be forwarded along the shortest path. Figure 3.2 illustrates the stretch of alternate paths of different topologies. To ensure an unbiased evaluation, only paths between node pairs with existing LFAs or E-LFAs are taken into consideration. OSPF re-route paths are best possible paths that avoid the failed elements (local links or local nodes). The graphs illustrate inverse cumulative distribution function of the stretch of E-LFAs in comparison with OSPF re-route.

From simulation results, the averages of optimal stretch of considered paths across all topologies are 1.237 for link protection and 1.213 for node protection while the average stretch of paths via LFAs and E-LFAs are 1.257 under LFC and 1.252 under NPC.

In addition to stretch, the number of hops of an alternate path also reflects the link utilisation. Figure 3.3 illustrates the numbers of hops forming alternate paths. The comparison is made between OSPF re-route for link protection and E-LFAs under LFC, and OSPF re-route for node protection and E-LFAs under NPC.

The average numbers of hops of OSPF re-route are 4.731 for link protection and 4.744 for node protection while paths via LFAs and E-LFAs have average numbers of hops equal to 4.741 under LFC and 4.750 under NPC. Although the results of the optimal shortest paths are not plotted due to different number of paths being considered (*i.e.* LFAs and E-LFAs do not exist for all node pairs), the average number of hops of the shortest paths across all topologies is 4.271.



(a) Abilene (b) GEANT (c) Abovenet

(d) Sprintlink (e) Tiscali (f) Waxman

(g) Barabasi-Albert (h) Barabasi-Albert-2

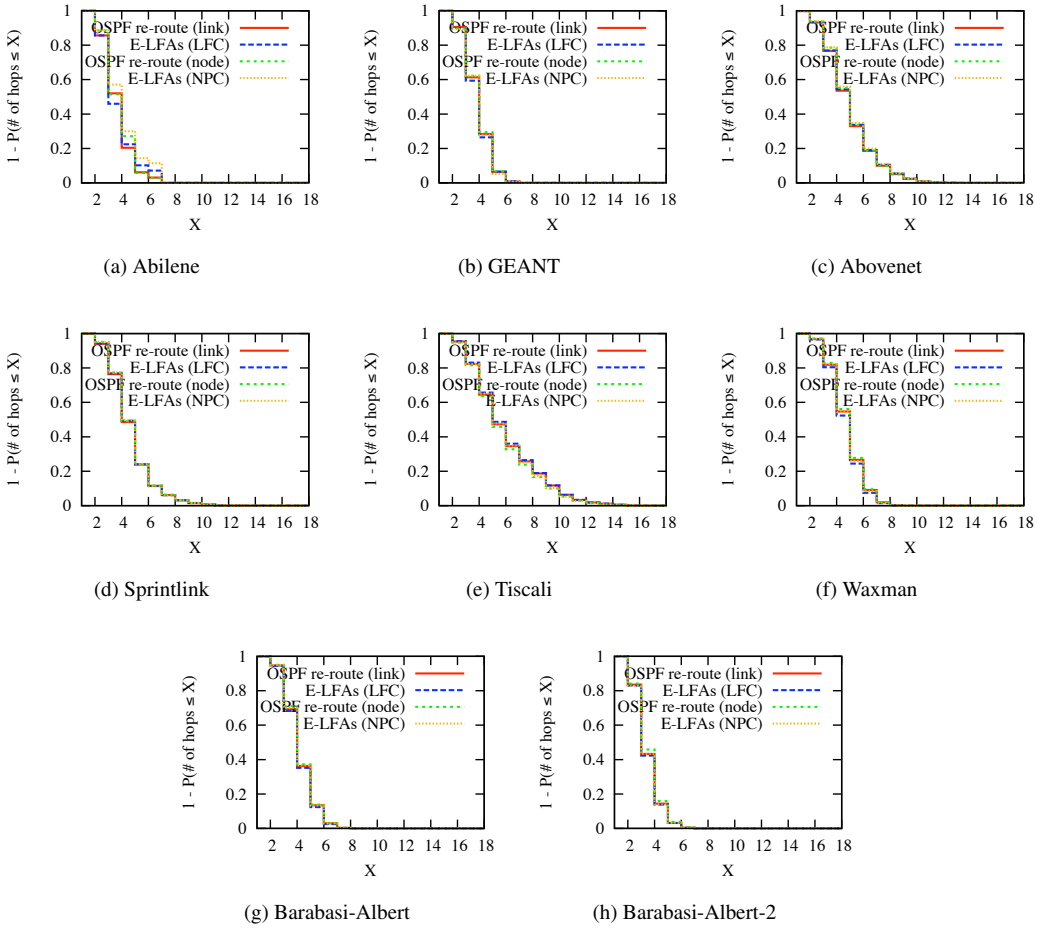
Figure 3.2: Stretch comparison between OSPF re-route and E-LFAs.

3.7 Conclusions

This chapter presented a technique called the Enhanced Loop-Free Alternates (E-LFAs) to elevate the repair coverage provided by Loop-Free Alternates (LFAs) [8]. Depending on the preference of network operators, a router can recover packets from either *local link* or *local node* failures based on different conditions.

E-LFAs are determined by using a simple recursive method on the normal LFAs. If a neighbour is not qualified as LFA of the detecting node, its LFA is considered instead. The process iterates until a node that is capable of forwarding packets to the destination without traversing the failed element is found. Consequently, the neighbour of the detecting node starting that path can be assigned as an E-LFA. Furthermore, additional information about the number of times a re-routed packet must traverse through the alternate next hops via E-LFA until it can be forwarded along the shortest path again must be stored.

E-LFAs inherit the loop-free property of LFAs, which was proved in Section 3.5. The simulation results showed that by employing E-LFAs, the repair coverage can be improved up to 7.758% for link protection and 13.547% for node protection in average relative to a 100% of protectable destinations. Furthermore, the path length stretch and the overheads of E-LFAs are considerably low. It is greatly



(a) Abilene (b) GEANT (c) Abovenet

(d) Sprintlink (e) Tiscali (f) Waxman

(g) Barabasi-Albert (h) Barabasi-Albert-2

Figure 3.3: Number of hops of a path before and after failures under OSPF re-route and E-LFAs.

believed that E-LFAs are good alternatives for improving network reliability without sacrificing the performance of normal routing or increasing the complexity in the management plane.

Chapter 4

Achieving Full Fast Failure Recovery

4.1 Introduction

The concept of fast re-route is specified in the IP Fast Re-Route (IPFRR) framework [113] and discussed in Section 2.3.3. However, it is clearly illustrated that a simple technique such as Loop-Free Alternates (LFAs) [8] cannot guarantee full protection against single failures. Chapter 3 introduces a technique called Enhanced Loop-Free Alternates (E-LFAs) to elevate the repair coverage of normal LFAs. However, providing a near optimal coverage is not sufficient in a network operator point of view. Therefore, it is important to design a routing strategy which is capable of fully handling single (either link or node) failures without jeopardising the network performance with respect to other metrics (*e.g.* router performance and network traffic).

This chapter presents a novel approach called Full Fast Failure Recovery (F3R). It is based on an approach similar to tree colouring using a simple shortest path algorithm to build two disjoint trees. Basically, F3R provides end systems access to two disjoint trees: *red* and *blue*. The routing technique enhances the information about the next hop and its complement to permit full protection against single link failures. Furthermore, F3R allows a router to perform traditional forwarding process. That is, only destination address is required to forward the packets.

In general, F3R is considered an IPFRR technique as it provides fast re-route mechanism using pre-computed paths. In other words, when a link fails, another path which is the complementary path is used for packet forwarding. This allows a router to suppress the global advertisement of transient failures, which eliminates the routing instability problem due to redundant convergence processes[1]. To permit a consistent routing, the failure-detecting node must indicate the packet as being "re-routed" so that other nodes can determine a path the packet must traverse. Explicit notification of failure is not essential and other types of signalling is not required to fast re-route a packet under F3R. Once the failure is recovered, the packets are forwarded via the normal paths. Nevertheless, an exception is made when a failure persists longer and the detecting router must trigger the re-convergence process. When this occurs, all routers must re-calculate both the first and second sets of paths.

The deployment of F3R technique is simple for network employing a link-state routing protocol such as OSPF or IS-IS as each node has the global knowledge of the underlying network topology.

[1] It has been observed that most failures are transient, most of which last less than a minute [52, 73].

Furthermore, routing under F3R guarantees a loop-free environment as all routers share a consistent view of routing paths. The followings summarise the main features of F3R:

- It provides different loop-free paths for routing in normal and different failure scenarios.
- It guarantees a 100% repair coverage against any single link failures and eliminates routing instabilities caused by redundant failure notifications, which trigger the network to re-converge.
- It requires minimum changes to traditional IP routing as the only requirements are packet marking and an additional information about the next hop for each existing routing table entry in the routing table.

4.2 Disjoint Trees

In general, the paths between any node pairs in the network are computed based on the weights of links forming them. Routing protocols such as OSPF and IS-IS employ link-state algorithm; hence, each router has a global knowledge of the network map. Consequently, the least cost paths are chosen for packet forwarding to ensure that a routing protocol utilises the network resources efficiently. Nevertheless, link weights do not always imply the actual physical distances as it is often determined and explicitly set by network operators.

Constructing two disjoint trees rooted at the same destination for an arbitrary network can elevate its reliability. That is, one tree is used for normal routing and the other for routing under failure scenarios. However, it is not always possible to find a tree that is fully disjoint with the shortest path tree in the same network topology. Therefore, migrating out of the shortest path paradigm is one of the simplest solutions to permit disjoint trees approach. This is also known as tree colouring where one tree is indicated as *red* and another as *blue*. Some technique uses tree colouring to handle dual link failures [60] given that all elements in the network are three-edge-connected. Although F3R handles only single link failure, it does not pose any requirements on the network topology. Furthermore, the technique works for an arbitrary network.

4.3 Computing Red Trees

Intuitively, disjoint trees rooted at the same destination given the same network graph can be found if both trees are created with regard to one another. More precisely, if either tree consists of at least one edge cut set[2], its disjoint does not exist.

Let $G = (V, E)$ be a directed graph with vertices $V = \{v_1, v_2, ...\}$ and edges $E \in V \times V$ representing the network topology. A weight $w(i, j) \in \mathbb{R} > 0$ is assigned to each edge (i, j). Given a node in V, n_i and a set of vertices being considered, C, a subgraph is defined as follows.

If $\{G_1, G_2, ...\}$ represents the set of connected graphs in $G - \{n_i\}$, $\{H_1, H_2, ...\}$ denotes the subgraphs of G from the nodes in $\{G_1 + \{n_i\}, G_2 + \{n_i\}, ...\}$.

[2] An edge cut set defines a group of edges whose total removal renders the graph disconnected.

Let $H(G, n_i)$ be the set of subgraphs, induced by n_i on G. If n_i is not a node cut set of size 1, then $H(G, n_i) = G$. Subgraphs are therefore, defined as any connected graphs induced by the removal of node n_i. Algorithm 4.1 computes the red tree rooted at the destination, d.

Algorithm 4.1 Computing the red tree rooted at d.

Input: d, G, $H(G', d)$

Output: $T_{red}(d)$

1: $G' \leftarrow G$
2: $T_{red}(d) \leftarrow ShortestPath(G', d)$
3: $C \leftarrow d$
4: **while** $C \neq \emptyset$ **do**
5: $n_i \leftarrow \phi(C)$
6: $C \leftarrow C - \{n_i\}$
7: **for all** $H \in H(G', n_i)$ **do**
8: $N \leftarrow n(H, n_i)$
9: $min\_node \leftarrow x : min_{x \in n(H,n_i)}(w(x, n_i))$
10: **for all** $x \in n(H, n_i) - \{min\_node\}$ **do**
11: $E' \leftarrow E' - (x, n_i)$
12: $E' \leftarrow E' - (n_i, x)$
13: **end for**
14: $C \leftarrow C \cup \{min\_node\}$
15: $T_{red}(d) \leftarrow ShortestPath(H, d)$
16: **end for**
17: **end while**
18: **return** $T_{red}(d)$

Each subgraph contains a set of nodes, $n(H, n_i)$, adjacent to n_i. The algorithm reduces the size of $n(H, n_i)$ to 1 by removing all links except for the one with a minimum weight. The links being removed are considered *redundant*. The node connected by the remaining link, min_node is added to C. Note that, $\phi(C)$ implies the first element in C. After that, $T_{red}(d)$ is adjusted using the shortest path algorithm (except when the size of $n(H, n_i)$ is equal to 1). The process iterates until no more nodes need to be considered (*i.e.* C becomes empty). Upon the completion of the algorithm, $T_{red}(d)$ can be used for either in normal or failure cases.

To elaborate the computation of red tree, a simple network topology illustrated in Figure 4.1 is used as an example. From Figure 4.1, bold arrows form a shortest path tree rooted at R6. This tree is used to initialise $T_{red}(R6)$ for red tree computation given the destination is R6.

The algorithm first determines the immediate child nodes of R6 in $T_{red}(R6)$, R4 and R5. Since R4 is connected to R6 through a minimum weight link, link R5—R6 is removed. After that, $T_{red}(R6)$ is adjusted based on the shortest path algorithm (Figure 4.2b). Link R4—R6 becomes a part of the final red

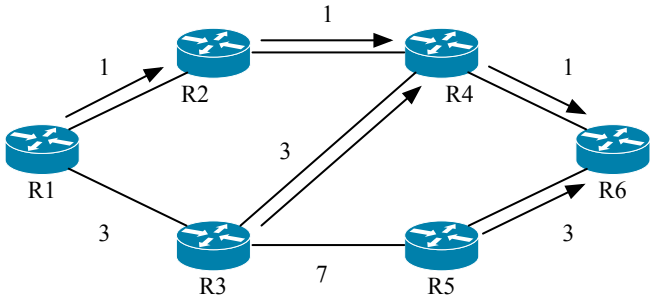
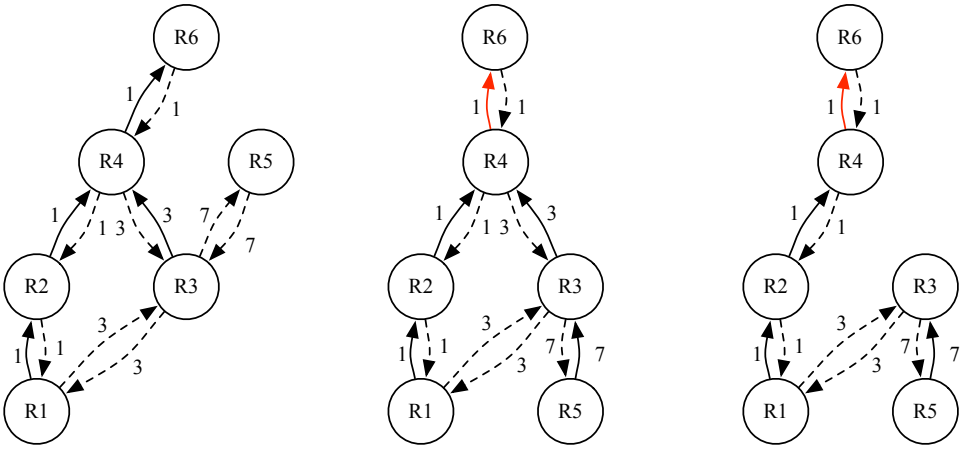


Figure 4.1: A simple network topology used for F3R illustration.

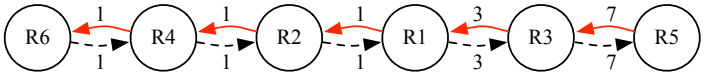
tree. Next, the immediate neighbours of R4 are determined. As link R2—R4 has a minimum weight, link R3—R4 is removed and $T_{red}(R6)$ is adjusted similarly as in the previous step. The remaining nodes: R2, R1, R3, and R5 are yet considered. However, only one immediate child node belongs to each of these nodes; hence, no further action is required.



(a) Remove R5—R6.

(b) Adjust the SPT.

(c) Remove R3—R4.



(d) Red tree rooted at R6.

Figure 4.2: Computation of red tree rooted at R6.

Figure 4.2d illustrates the final red tree, $T_{red}(R6)$ used for routing tables construction. It can be seen that $T_{red}(R6)$ does not consist of any set of edges that forms an edge cut set in the network graph (given that the graph is directed).

4.4 Computing Blue Trees

After the red tree, $T_{red}(d)$ is constructed, its corresponding blue tree, $T_{blue}(d)$ can be found by re-calculating the link weights in the directed graph, G.

Let $E(T_{red}(d))$ be a set of links in G employed by $T_{red}(d)$. The simplest method that can be used to compute $T_{blue}(d)$, is to perform a shortest path algorithm on a network graph G with $E(T_{red}(d))$. However, G is often rendered disconnected when all links in $E(T_{red}(d))$ are completely removed. This is due to the unpredictable formation of an arbitrary network (*i.e.* G may consist of one or more single degree nodes). Thus, F3R increases the weights of links in $E(T_{red}(d))$ by a very large value so that traversing through all links in the network costs less than passing through a single link in $E(T_{red}(d))$. If a link in $E(T_{red}(d))$ is still employed in the $T_{blue}(d)$, it implies that there is no disjoint element for that particular link in G. The large value used to compute blue trees is defined as the sum of weights of all links in E:

$$W_t := \sum_{(i,j)\in E} w(i,j) \tag{4.1}$$

Using W_t as a link weights re-calculation factor, Algorithm 4.2 computes the blue tree rooted at the destination, d.

Algorithm 4.2 Computing the blue tree rooted at d.

Input: d, G, $E(T_{red}(d))$

Output: $T_{blue}(d)$

1: $G' \leftarrow G$

2: **for all** $(i,j) \in E'(T_{red}(d))$ **do**

3: $\quad w(i,j) \leftarrow w(i,j) + W_t$

4: **end for**

5: **return** $T_{blue}(d) \leftarrow ShortestPath(G', d)$

Consider the network shown in Figure 4.1. $T_{red}(R6)$ is obtained in Section 4.3. Using Equation 4.1, W_t of the network is equal to 19. The algorithm first increases the weights of all links in $E(T_{red}(R6))$ by 19. After that, F3R runs the shortest path algorithm on the re-calculated set of links to obtain $T_{blue}(R6)$, which is directed disjoint from $T_{red}(R6)$. Figure 4.3 illustrates the final blue tree, $T_{blue}(R6)$.

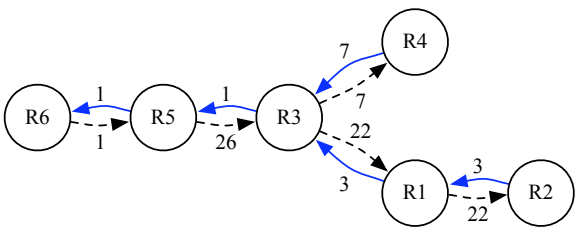


Figure 4.3: Blue tree rooted at R6.

It is important to note that, all links considered in Algorithm 4.1 and Algorithm 4.2 are assumed to be *directed*. However, F3R can be employed in both directed and undirected network graphs to fully protect all recoverable single link failures.

4.5 Packet Processing and Forwarding

Under F3R routing strategy, each router can forward packets in accordance to two disjoint trees, a packet must be marked to retain the routing consistency throughout the network. In F3R, neither tree offers optimal routing paths to all destinations. That is, even if in failure-free case, least cost paths may not be employed. Nevertheless, deploying F3R allows network operators to overcome the forwarding discontinuation during network re-convergence. Let `in_pkt` be the incoming packet and `out_pkt` be the outgoing packet to node d, arriving at node s, Algorithm 4.3 summarises the packet processing at node s.

Algorithm 4.3 Packet processing at node s.

Input: `in_pkt`

Output: `out_pkt`

if $\texttt{in\_pkt}.R == 0$ **then**
  if $(s, n_{\texttt{in\_pkt}.T}(s, \texttt{in\_pkt}.d)) ==$ failed **then**
    $\texttt{in\_pkt}.R \leftarrow 1$
    $\texttt{in\_pkt}.T \leftarrow \texttt{in\_pkt}.Inverse(T)$
    return `out_pkt` $\leftarrow$ `in_pkt`
  else
    return `out_pkt` $\leftarrow$ `in_pkt`
  end if
else
  if $(s, n_{\texttt{in\_pkt}.T}(s, \texttt{in\_pkt}.d)) ==$ failed **then**
    Drop(`in_pkt`)
    return null
  else
    return `out_pkt` $\leftarrow$ `in_pkt`
  end if
end if

Based on similar operations, two different options for marking re-routed packets under F3R are proposed as follows:

R-bit Marking Scheme

This marking scheme allows packets to be forwarded normally without any marking. In the presence of a failure, the detecting node sets R-bit to indicate a re-routed packet. To employ this scheme, either red or blue tree must be indicated as *primary* and the other, *secondary*. In general, the tree that provides

a greater number of shorter paths is set as primary to provide as many efficient paths as possible. The complementary tree is used to re-route packets in case of failures.

R-bit/T-bit Marking Scheme

This marking scheme allows packets to be forwarded along the shorter path provided by two disjoint trees. However, it requires routers to consider traffic engineering awareness in the failure-free cases. Under this marking scheme, R-bit indicates whether the packet has yet experienced the failure. This bit is initially set to 0 by the packet originating router. However, the router must set T-bit to identify the routing tree that provides a shorter path given the same destination. When a failure occurs, the detecting node inverts both bits and forwards the packets according to the other routing tree (*i.e.* red or blue).

If only R-bit is employed, line 4 of Algorithm 4.3 can be omitted while $n_{\texttt{in\_pkt}.T}(s, \texttt{in\_pkt}.d)$ in lines 2 and 10, represent the next hop link in the primary and secondary routing trees respectively. For deployment in IP networks, it is recommended that few bits in the Type of Service (TOS) field in IPv4 packet or the Traffic Class field in IPv6 packet are used.

4.6 Suppressing Failure Notification

The aim of F3R is to reduce the packet loss rates during network re-convergence in particular for those caused by intermittent failures. F3R provides a routing strategy that can re-route packets whenever a single link failure occurs; hence, re-converging the network is not necessary unless the failure persists for a certain duration. Routing under F3R suppresses the failure notification for a minute to reduce unnecessary instabilities and forwarding discontinuation. However, in case of persistent failures, the convergence process is eventually triggered and both trees (*i.e.* red and blue) must be re-computed.

4.7 Optimisation

As can be seen in Algorithm 4.1 and Algorithm 4.2, the computations of both trees heavily depend on the shortest path algorithm. Furthermore, extracting the subgraphs of a network also relies on the shortest path algorithm for simplicity. Consequently, the complexity of F3R increases proportionally to the number of shortest paths finding operations. Therefore, F3R employs the incremental Shortest Path First (iSPF) algorithm [75] to reduce its computation time. That is, the shortest path algorithm needs to be run on affected parts of the graph only. Moreover, this computation can be performed by a number of routers in large networks in a distributed fashion to speed up the routing tables construction.

4.8 Properties

The two key properties of F3R are: *a)* full repair coverage for recoverable single link failures and *b)* loop-free forwarding. First, Theorem 4.1 shows that routing under F3R is complete under both normal and failure scenarios.

Theorem 4.1. *Given a source s, a destination d, its next hop in $T_{red}(d)$, $n_{red}(s,d)$, and its next hop in T_{blue}, $n_{blue}(s,d)$. If G is not disconnected after the removal of link $(s, n_{red}(s,d))$, then there exists a path from s to d via $n_{blue}(s,d)$ that does not traverse link $(s, n_{red}(s,d))$ under fast re-route using F3R.*

Proof. Let $\mathcal{D}$ be the set of paths from s to d that do not include $(s, n_{red}(s,d))$. If $\mathcal{D} = \emptyset$, the failure is non-recoverable. Thus, the proof proceeds to ensure that when $\mathcal{D} \neq \emptyset$, F3R always finds an alternate path from s to d in $\mathcal{D}$ successfully.

For each subgraph, H in $H(G', d)$, all links are categorised into: *a)* avoidable components, E_a, and *b)* unavoidable components, E_u. E_a contains a set of links which can be removed from $T_{red}(d)$ without disconnecting H while E_u contains a set of links that can be used only if an alternate path from s to d does not exist. For each node in $T_{red}(d)$, F3R eliminates all links in E_a leaving only E_u in $T_{red}(d)$. Thus, no links in E_u can be removed without rendering the graph disconnected.

F3R constructs $T_{blue}(d)$ with regard to $T_{red}(d)$ using W_t as a link weights re-calculation factor. Since each link in E_u is increased by W_t, no links in E_u is employed in $T_{blue}(d)$ given that $\mathcal{D} \neq \emptyset$.

Let $E(T_{red}(d))$ be a set of links in $T_{red}(d)$ and $E(T_{blue}(d))$ be a set of links in $T_{blue}(d)$. As $E(T_{red}(d)) \cap E_u = E(T_{red}(d))$ and $E(T_{blue}(d)) \cap E_u = \emptyset$, $E(T_{red}(d)) \cap E(T_{blue}(d)) = \emptyset$. This implies that $T_{red}(d)$ and $T_{blue}(d)$ are disjoint trees. Therefore, if link $(s, n_{red}(s,d))$ fails, a packet can be forwarded to d via $(s, n_{blue}(s,d))$. □

Theorem 4.2. *If there exists a path from s to d without link $(s, n_{red}(s,d))$, fast re-route using F3R can forward packets from s to d without traversing $(s, n_{red}(s,d))$.*

Proof. Let $P_{red}(s,d)$ be the path in $T_{red}(d)$ and $P_{blue}(s,d)$ be the path in $T_{blue}(d)$ from s to d. By definition, a tree is a connected graph without cycles. Thus, routing along a single tree either $T_{red}(d)$ or $T_{blue}(d)$ does not cause a forwarding loop.

If link $(s, n_{red}(s,d))$ fails, a packet can be re-routed via link $(s, n_{blue}(s,d))$ to d successfully as $T_{red}(d)$ and $T_{blue}(d)$ are disjoint. □

4.9 Performance Evaluation

This section evaluates the performance of F3R routing strategy. Similar to Chapter 3, overheads incurred by the routing technique and the repair coverage are considered. Since F3R does not employ the optimal shortest paths under normal scenario, a comparison between F3R paths and the shortest paths is also given.

4.9.1 Method

This chapter employs similar method used in Section 3.6 for evaluation of alternate paths characteristics. However, extensions to the Java software are made to permit F3R paths computation. These extensions are verified as described in Section 3.6. A machine with 2.16 GHz Intel Core 2 Duo processor and 2 GB memory is used throughout the evaluation. The evaluation of F3R employs the same set of topologies used in Section 3.6.

4.9.2 Overheads

The followings evaluate different types of overheads:

Computational Overhead

The computation time required for completing F3R algorithm is divided into: *a)* the time required to compute red trees and *b)* the time required to compute blue trees. The expected results of *b)* are trivial as only a linear operation and iSPF need to be performed. Assuming the computation is not distributed, the time complexity incurred on each router to calculate blue trees for all destinations can be at worst, represented by $O(V^3 + E \times V)$ where V is the number of vertices and E is the number of edges. This notation assumes a simple priority queue implementation of Dijkstra's algorithm, where its complexity is $O(V^2)$. If a Fibonacci heap is implemented, the time complexity can be reduced to $O(E^2 + V^2 \times \log V)$ given that the complexity of the heap is $O(E + V \times \log V)$. The results show that given the largest backbone topology used in simulation (*i.e.* Sprintlink), the time to compute blue trees for all destinations on each router is less than 100 ms which is considerably low.

On the contrary, the empirical results of *a)* show that as the network size grows, the amount of time required to compute red trees increases significantly. For small networks such as Abilene and GEANT, the average of computation times of red trees for all nodes in the network are 0.459 ms and 1.276 ms respectively. Nevertheless, for networks consist of more than 100 nodes, the computation can be very slow. For examples, the average computation times of red trees of the Abovenet and Tiscali are more than 11 minutes and 23 minutes, respectively. Recall that, for each node in the shortest path tree rooted at the destination being considered, subgraphs of the tree below that node need to be decomposed. This process incurs a very long computation time. For a strongly connected network (*i.e.* all elements are recoverable), this process can be omitted and the time required to compute red trees can be considerably shorter. More precisely, the computational complexity required to compute red trees for all nodes in the topology can be represented by $O(V^5 \times E)$.

As a result, F3R is recommended for networks smaller than 100 nodes or larger strongly connected networks (where the decomposition of subgraphs can be omitted).

Memory Overhead

Similar to other IPFRR techniques that employ pre-computed paths, F3R requires a router to enhance the existing routing table entries with additional information about the alternate next hops. This requirement is similar to that of multi-topology approach, but limit the number of next hops required per destination to 2.

Packet Overhead

Refer to Section 4.5, F3R requires 1 bit for R-bit and 2 bits for R-bit/T-bit marking scheme to provide fast re-route in case of single link failures. Unlike E-LFAs, the packet overhead of F3R is not topology dependent. Furthermore, no additional bit is required to guarantee a loop-free forwarding due to the characteristics of disjoint trees. These bits can be stored in Type of Service (TOS) field in IPv4 packet or Traffic Class field in IPv6 packet.

4.9.3 Repair Coverage

The repair coverage of F3R is proven in Section 4.8. It guarantees a 100% coverage for all protectable destinations under any single link failure scenarios.

4.9.4 Stretch

The path length stretch is one of the most important factors that determines the feasibility of a resilient mechanism. If the paths obtained have too high stretch, the impact on network traffic can be severe. Figure 4.4 shows that the stretch of F3R paths is very high in relative to the optimal shortest paths. The graphs illustrate inverse cumulative distribution function of the stretch of F3R in comparison with OSPF re-route. The average of optimal stretch is 1.221 while the average stretches of normal and alternate paths provided by F3R (under R-bit/T-bit marking scheme) are 1.258 and 11.153. Interestingly, it can be clearly seen that, although not directly proportional, larger networks often incur significantly higher stretch. This is due to the characteristics of disjoint trees, which either one has a very inefficient path towards the same destination.

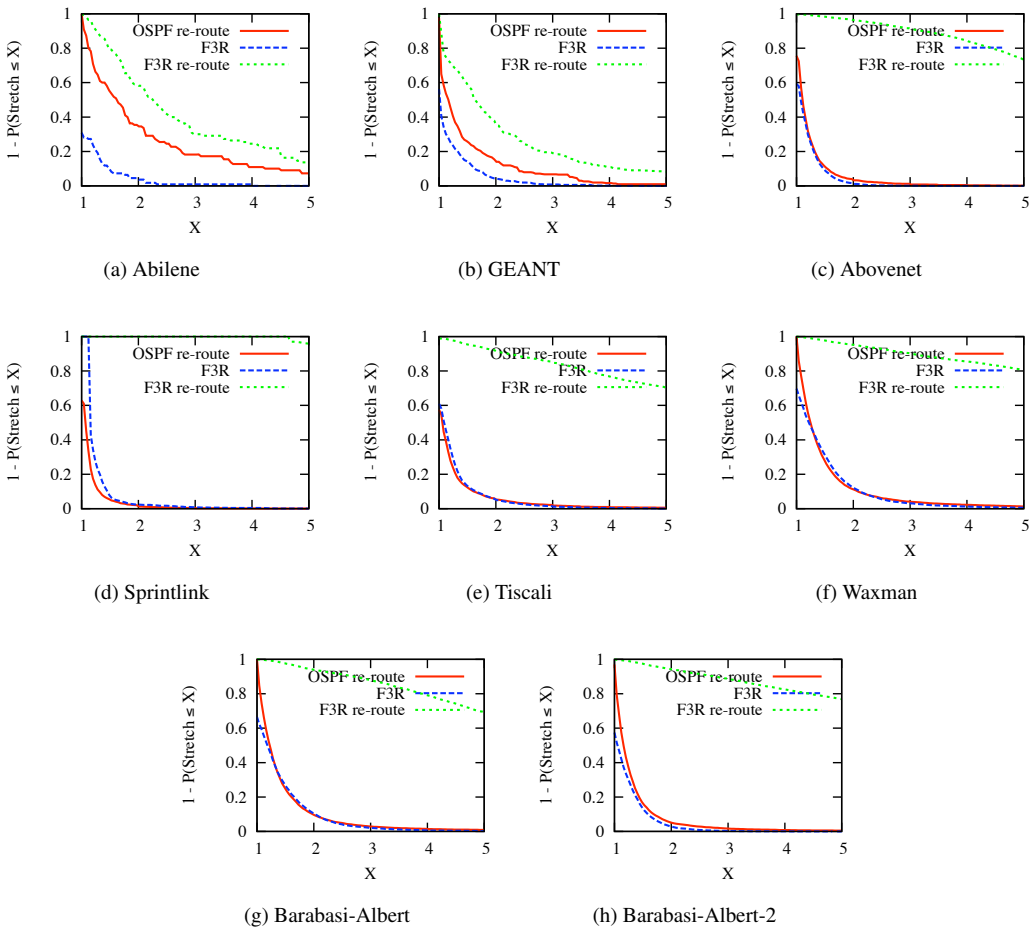


Figure 4.4: Stretch comparison between OSPF re-route and F3R under normal and failure cases.

In most cases, the stretch and the number of links used in a path are directly proportional (*i.e.* it is likely that the larger number of hops a path requires, the higher stretch it incurs). However, when the

stretch is near optimal, it is important to incorporate the number of hops in the evaluation to ensure the optimality. The simulation results of F3R show that, no further evaluation must be made as it is clearly seen that, the stretch incurs by the routing strategy is significant. Furthermore, this trivially incurs a huge impact on network load in particular for large topologies.

In general, a simple method can be used to test the impact on network load. Under same failure scenarios, the traffic matrix is injected into the network so that it saturates most links after the network re-converges. After that, the traffic characteristics such as the Maximum Link Utilisation (MLU) and the total network overhead are measured. If the values of these parameters under an IPFRR technique far exceed those of normal convergence, then the impact on network is considered very high. For routing under F3R, high path length stretch reflects poor link utilisation; hence, if the network is saturated by traffic matrix injection, most links are potentially overloaded.

4.10 Conclusions

This chapter introduced a new IPFRR technique based on disjoint trees called Full Fast Failure Recovery (F3R). The technique can re-route packets from all recoverable single link failures. It aimed to alleviate the packet loss rates resulted from forwarding discontinuation during network re-convergence. Although it is similar in spirit to the tree-colouring-based technique such as [60], F3R does not make any assumption on the formation of underlying network topology. That is, it can be deployed in an arbitrary network.

The technique has two key properties which are completeness and correctness. These properties were proved mathematically. An extensive evaluation was made mainly based on paths characteristics. However, the protocol overheads were also considered.

F3R's computational overhead is considerably high depending on the size of a network. It was observed that F3R is practical for deployment in networks that have less than 100 nodes. The approach may be feasible in larger networks only if they are strongly connected as the computation time can be reduced by omitting the subgraphs decomposition process. Furthermore, the computations of disjoint trees may be distributed performed to reduce computational processing on individual router. On the contrary, the memory and packet overheads are considerably low. Nevertheless, the paths employed during failure scenarios have a significantly higher stretch than that of optimal shortest paths upon the completion of re-convergence process. More precisely, larger networks incur higher stretch paths under F3R routing technique. For smaller topologies such as Abilene and GEANT, the F3R paths seem to be practical.

Chapter 5

Alternate Next Hop Counting Mechanism

5.1 Introduction

In general, computing backup paths that can be used to deliver packets correctly can be very simple and optimal if the failures are globally known immediately after they occur. However, this is not possible due to unavoidable propagation delay of failure advertisements. One way to allow consistent routing using alternate paths is to employ source routing with a risk of a router performance being degraded as well as the incurrence of security concerns. These include the advertisements of falsified route information and Denial-of-Service (DoS) attacks [106]. Designing a hop-by-hop routing protocol that can recover from failures quickly while keeping the administrative complexity to a minimum is difficult. It is mainly because of inconsistent network map among routers during network re-convergence, which often leads to packet loss and forwarding loops. Consider a network topology in Figure 5.1, if link R5→R6 fails, the most conventional method for R5 to find a new route is to re-calculate the shortest paths for affected destinations[1] based on the updated link-state database[2]. However, the knowledge of the new shortest path R5→R2→R1→R4→R6 is locally known only to R5 before it generates and sends any failure notification messages to other nodes. Consequently, packets destined for R6 routed via R5 may be caught in a forwarding loop or dropped (if their TTL expires) until all intermediate routers update their routing tables.

Alternate next hop counting is a novel mechanism used in conjunction with specific backup path computation to provide fast re-route in traditional IP networks [69]. Re-routing in this technique relies on the Alternate Next Hop Counter (ANHC) in the packet header to ensure correct forwarding. After the normal shortest paths are calculated, a router first determines the sum of all link weights W_t. For each node pair (s, d), a router adds W_t to the weights of the links present in the shortest path between s and d and re-calculates the shortest path from s to d. This *secondary path* is used to calculate the *alternate next hop* that s will use to send packets to d if the shortest path becomes unavailable. To allow a consistent routing only on the basis of alternate next hops, each router sets the ANHC value for every destination equal to the number of times the packet needs to be forwarded using the alternate next hops. If the ANHC holds a positive number, a router decreases its value by 1 and forwards the packet to the

[1] For a network shown in Figure 5.1, only R6 is affected when link R5→R6 fails.

[2] Note that this process is part of the basic specification of current routing protocols such OSPF and IS-IS.

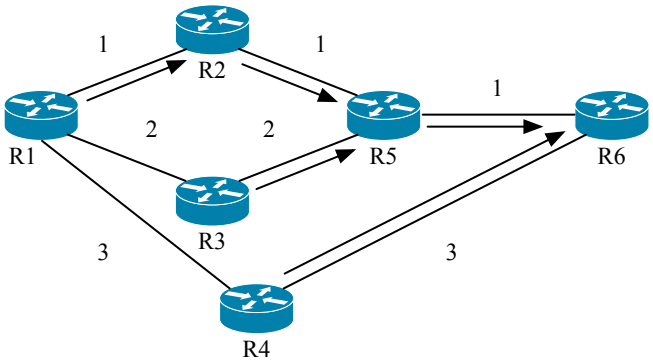


Figure 5.1: A simple network topology illustrating shortest paths to R6.

alternate next hop corresponding to its own secondary path to d. If the ANHC value equals 0, the router forwards the packet via its normal path to d (the next hop according to the shortest path route). The main features of fast re-route using ANHC are summarised as follows:

- It is based on the normal shortest path algorithm, which can be easily implemented and deployed without any major modifications to existing routing standards. That is, no additional mechanisms such as tunnelling and source routing are required.
- It offers optimal paths in normal scenarios and the corresponding loop-free alternate paths in the presence of single link failures. This can significantly reduce the amount of packet loss due to network re-convergence.
- It guarantees a 100% repair coverage for any single link failures and eliminates routing instabilities caused by redundant failure notifications, which trigger the network to re-converge.

5.2 Computing Alternate Paths

To employ an IPFRR technique such as LFAs, a router must be enhanced with additional information about the alternate next hop. Basically, the algorithm used to compute alternate paths or alternate next hops can be different from one routing strategy to another. Although fast re-route using ANHC uses different algorithm, its memory overhead is comparable to that of LFAs. This requirement is considered minimal given that a full protection is guaranteed.

Furthermore, designing an algorithm that suits the objectives is essential. For example, some algorithms may guarantee a backup path that can protect a node failure while the others can re-route packets successfully only if the root cause of failures is a fibre cut. One simple method described previously is to remove the next hop link from the network map (*i.e.* set the link weight in the adjacency matrix equals to infinity) and re-calculate the shortest paths. However, this method results in an inconsistent routing in traditional IP networks where the forwarding process relies only on the destination address.

Fast re-route using ANHC computes alternate next hops to protect local link failures. It calculates the backup path not only with regard to the next hop link but the entire path to each destination. This

creates a correlation among routes for all origins to the same destination. It is important to note that, this routing strategy is based on normal shortest path algorithm such as Dijkstra's algorithm; hence, the algorithm may fail due to network partitioning if all primary links are completely removed. Heuristically, the backup path computation is performed by increasing the weight of all links incorporate in the normal path with a very large value to ensure that the secondary path has as few links in common with the normal path as possible (*i.e.* maximally link disjoint path) without failing the algorithm.

Let $G = (V, E)$ be the graph with vertices $V = \{v_1, v_2, ...\}$ and edges $E \in V \times V$ representing the network topology. A weight $w(i, j) \in \mathbb{R} > 0$ is assigned to each edge (i, j). The total weight of all links in E is denoted by W_t as in equation 4.1.

Increasing the weight of a link by W_t is equivalent to replacing it with an infinite value. Let $E_p(s, d)$ be the set of links used in a normal path from s to d. The primary next hop and its alternate are denoted as $n_p(s, d)$ and $n_s(s, d)$. Using W_t as a link weight re-calculation factor, Algorithm 5.1 is run on each router for each source-destination pair. In Algorithm 5.1, the output of $ShortestPath$ is a shortest path tree T_s rooted at s, with $T_s(d)$ being the shortest path from s to d excluding s, and $\phi(T_s(d))$, the first node in $T_s(d)$. The alternate next hop from s to d, $n_s(s, d)$, can be found using the algorithm illustrated below.

Algorithm 5.1 Computing the alternate next hop.

Input: s, d, G, $E_p(s, d)$

Output: $n_s(s, d)$

1: $G' \leftarrow G$
2: **for all** $(i, j) \in E_p(s, d)$ **do**
3: $\quad w'(i, j) \leftarrow w(i, j) + W_t$
4: **end for**
5: $T'_s = ShortestPath(G', s)$
6: $H_s(s, d) \leftarrow T'_s(d)$
7: $n_s(s, d) \leftarrow \phi(T'_s(d))$
8: **return** $n_s(s, d)$

The outputs of Algorithm 5.1 over all node pairs are $S(d) = \{n_s(v_1, d), n_s(v_2, d), ...\}$, the alternate next hop nodes that every node uses to route packets to d under failure conditions, and $H_s(s, d) = \{h_1, h_2, ...\}$, where $H_s(s, d)$ is the backup path from s to d excluding s, and h_i represents the i-th next hop in the backup path from s to d. Thus, Algorithm 5.1 calculates an alternate next hop for each source-destination pair, which is the first hop of an alternate path that is maximally link disjoint from its corresponding normal path. For example, let R5 be the source and R6 be the destination of a network shown in Figure 5.1. The weight of link R5→R6 is increased by 13 (*i.e.* W_t). After that, R5 runs the shortest path algorithm based on the graph with new link weights and obtains the alternate next hop towards R6. Table 5.1 shows the alternate next hops from each node to R6.

Of course, each node $s \in G$ must calculate a different $n_s(s, d)$ for a given destination d. As a result,

Table 5.1: Next hops from each node to R6.

Node	**R1**	**R2**	**R3**	**R4**	**R5**
Normal next hop	R2	R5	R5	R6	R6
Alternate next hop	R4	R1	R1	R1	R2

alternate paths for a destination d are not consistent throughout the network. Fast re-route using ANHC is different from some other resilient mechanisms precisely because the alternate path locally known to each router does not have to be consistent. This implies that there is no guarantee that a path formed by concatenating the alternate next hops to a particular destination is loop-free. Thus, the technique needs to employ a mechanism that uses these pre-computed alternate next hops to route packets to their destinations via loop-free paths. A mechanism called *alternate next hop counting* is proposed to eliminate this path inconsistency problem.

5.3 Computing ANHC Values

As each router in the network may have different backup paths for the same destination, forwarding must be aided with an additional mechanism that allows correct operation under failure conditions. Fast re-route using ANHC uses alternate next hop counting to ensure that no forwarding loop is possible in the presence of a single link failure. The followings illustrate how, with inconsistent information on the local alternate paths, packets can be forwarded consistently under fast re-route using ANHC.

Alternate next hop counting makes use of an Alternate Next Hop Counter (ANHC) stored in the packet header. A few bits in the Type of Service (TOS) field in IPv4 packet or the Traffic Class field in IPv6 packet are sufficient to store the ANHC.

As defined in Section 5.1 for Algorithm 5.1, $S(d) = \{n_s(v_1, d), n_s(v_2, d), ...\}$, the $AHNC(s, d)$ value can be obtained using Algorithm 5.2.

The algorithm first considers the $current\_node$ which is initialised with s. The first hop in the alternate path from s to d is then compared with the alternate next hop of $current\_node$ to d. If they are the same node, $ANHC(s, d)$ is incremented by 1 and the alternate next hop of $current\_node$ to d becomes $current\_node$. After that, the second hop in the alternate path from s to d is used for comparison. This process iterates until either it reaches d or the condition fails, implying that it is no longer necessary to route via the alternate next hops. From this point, the packet can be forwarded normally. When the algorithm is terminated, $ANHC(s, d)$ is obtained.

Using the network shown in Figure 5.1, an example of the ANHC computation for the path from R5 to R6 (the alternate next hops from all nodes to R6 can be found for convenience in Table 5.1) is illustrated. Let R5 be the source and R6 be the destination. The shortest path from R5 to R6 is R5→R6 and its alternate path locally known to R5 is R5→R2→R1→R4→R6. For a node pair that has an alternate path, the condition in the first iteration of the algorithm is always true. The first hop of the alternate path from R5 to R6 is R2. The condition is true as R2 is also the alternate next hop of R5; hence, the value of ANHC is incremented by 1. Next, R5 considers the alternate next hop of R2, which is R1.

Algorithm 5.2 Computing the ANHC value.

Input: s, d, $H_s(s,d)$, $S(d)$

Output: $ANHC(s,d)$

1: $ANHC(s,d) \leftarrow 0$
2: $current\_node \leftarrow s$
3: $i \leftarrow 1$
4: **while** $h_i \neq d$ **do**
5: **if** $h_i == n_s(current\_node, d)$ **then**
6: $ANHC(s,d) \leftarrow ANHC(s,d) + 1$
7: $current\_node \leftarrow h_i$
8: $i \leftarrow i + 1$
9: **else**
10: **break**
11: **end if**
12: **end while**
13: **return** $ANHC(s,d)$

Since R1 is also the second hop of the alternate path from R5 to R6, the value of ANHC is incremented by 1. Similar result is obtained by considering R1; therefore, the value of ANHC becomes 3. As the last hop of the alternate path from R5 to R6 is different from the alternate next hop of R4 to R6, the loop is broken. The ANHC value for node pair from R5 to R6 is therefore, equal to 3. For simplicity in case of equal-cost paths in this example, the next hop with the lowest node number has the priority over the others. Table 5.2 shows the calculated ANHC value of each node pair in the network.

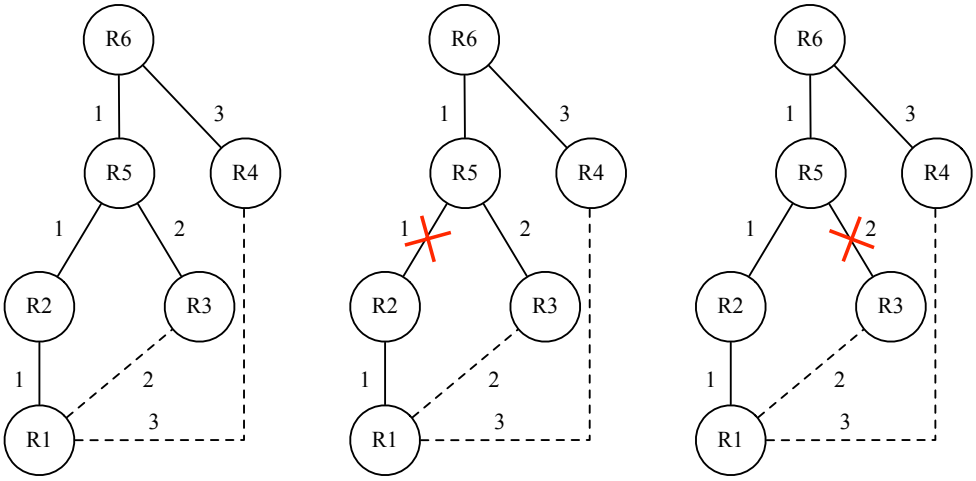
Table 5.2: The ANHC value of each node pair.

From/To	R1	R2	R3	R4	R5	R6
R1	-	2	2	2	1	1
R2	2	-	1	1	2	2
R3	1	1	-	1	1	2
R4	1	1	1	-	1	1
R5	1	1	1	2	-	3
R6	1	1	1	2	2	-

5.4 Computing ANHC Values Using Loop-Free Condition

In addition to Algorithm 5.2, there is an alternate algorithm known as recursive LFAs (rLFAs), which can be used to compute the ANHC values [69]. The main concept is based on a shortest path tree paradigm. Figure 5.2a illustrates the shortest path tree rooted at R6. If link R2–R5 fails (Figure 5.2b), packets

cannot be forwarded normally from any node within the same branch below the point of failures in the shortest path tree. That is, packets cannot be forwarded to R6 successfully from R2 or R1. However, once the packets are forwarded to another branch in the shortest path tree, routers can continue their normal routing operations without creating forwarding loops. For example, packets can be forwarded along the shortest path from R4 without traversing back to the point of failure, R2–R5. Another example is shown in Figure 5.2c, where packets can be forwarded to R6 normally once they reach R4.



(a) Shortest path tree rooted at R6. (b) Link R2–R5 fails. (c) Link R3–R5 fails.

Figure 5.2: Concept of recursive LFAs.

Recall the Loop-Free Condition (LFC) a node uses to determine its link-protection LFAs, ANHC values can be computed by utilising Equation 2.1.

Figure 5.2c is used to illustrate this alternate method for ANHC value computation. Since link R3–R5 fails, R3 cannot forward packets to R6 in the normal shortest path. After using Algorithm 5.1 to compute the alternate next hops for all node pairs, the corresponding ANHC value from R3 to R6 is computed as follows. First, R3 determines its alternate next hop, R1, using LFC. As the distance from R1→R6 is shorter than that of path R1→R3→R5→R6, the computation terminates and the ANHC value from R3 to R6 remains as 1. This implies that, when R3 detects a failure while sending packets to R6, it needs to forward the packets to its alternate next hop, R1, before these packets can be forwarded along the normal shortest path again.

Algorithm 5.2 and Algorithm 5.3 share the same properties, which are described later in this chapter. In addition, they require identical computational complexity and do not incur any significant difference in terms of overheads. The simulation results are generated based on Algorithm 5.2 without any comparison with Algorithm 5.3 as their re-routing paths are almost identical.

5.5 Packet Processing and Forwarding

Since the alternate next hop counting mechanism does not affect normal route calculation, packets can be forwarded to all destinations via the shortest paths in the absence of failures. When a node s detects

Algorithm 5.3 Computing the ANHC value using LFC.

Input: s, d, $H_s(s,d)$, $S(d)$

Output: $ANHC(s,d)$

1: $ANHC(s,d) \leftarrow 1$
2: $current\_node \leftarrow s$
3: $C = \emptyset$
4: **while** $current\_node \neq d || current\_node \notin C$ **do**
5: $C \leftarrow C \cup current\_node$
6: **if** $cost(current\_node, d) < cost(current\_node, s) + cost(s, d)$ **then**
7: **break**
8: **else**
9: $ANHC(s,d) \leftarrow ANHC(s,d) + 1$
10: $current\_node \leftarrow n_s(current\_node, d)$
11: **end if**
12: **end while**
13: **return** $ANHC(s,d)$

a failure in one of its outgoing links, it marks those packets which would be forwarded through the affected link with the ANHC value corresponding to the destination router of the packet. If an alternate path to that node exists, it decrements the ANHC value in the packet header and forwards the packet to its alternate next hop. When a node receives a marked packet, it determines the value of ANHC. If ANHC holds a positive number, its value is decremented by 1 and the packet is forwarded to the local alternate next hop. However, if the ANHC value is 0, the node forwards the packet to its normal next hop until it reaches the destination. Algorithm 5.4 summarises the operations when a packet arrives at each node.

The followings illustrate the packet processing and forwarding operations at each node under fast re-route using ANHC. Let R6 be the destination. When link R5→R6 fails, R5 detects the failure. Any packets being forwarded to R6 via R5 need to be re-routed. To avoid unnecessary re-convergence episodes for transient failures, failure advertisements are suppressed for a given duration. Instead, packets for affected destinations are re-routed using backup paths.

First, R5 determines whether the ANHC value of the packet to be routed is 0. If it is, the packet is forwarded normally (to its shortest path next hop) unless the next hop would be reached through a failed link. In this case, the router sets the ANHC based on the destination router for this packet. The ANHC value from R5 to R6 is 3. The node thus forwards the packet to its alternate next hop, R2, decrementing the ANHC value to 2 in the process and updating it on the packet. Once R2 receives the packet, it determines that since ANHC > 0, it must be decremented to 1 and updated on the packet, which is forwarded to R1 (its alternate next hop). The packet arrives at R1 and since again ANHC > 0, it goes through the same process and reaches R4. When R4 receives the packet, the ANHC value is 0; therefore,

Algorithm 5.4 Packet processing at node s.

Input: `in_pkt`

Output: `out_pkt`

if `in_pkt`.$ANHC == 0$ **then**
  if $(s, n_p(s, \texttt{in\_pkt}.d)) ==$ failed **then**
    `in_pkt`.$ANHC \leftarrow ANHC(s, \texttt{in\_pkt}.d) - 1$
    return `out_pkt` $\leftarrow$ `in_pkt`
  else
    return `out_pkt` $\leftarrow$ `in_pkt`
  end if
else
  if $(s, n_s(s, \texttt{in\_pkt}.d)) ==$ failed **then**
    Drop(`in_pkt`)
    return null
  else
    `in_pkt`.$ANHC \leftarrow$ `in_pkt`.$ANHC - 1$
    return `out_pkt` $\leftarrow$ `in_pkt`
  end if
end if

R4 forwards the packet to the destination via its normal next hop.

It is important to note that fast re-route using ANHC handles only *single* link failures. Certain cases of multiple failures can lead to forwarding loops. For example, when link R5→R6 fails, a re-routed packet is forwarded via the path R5→R2→R1→R4→R6. However, if link R4→R6 fails at the same time the packet arrives at R4, it is not discarded since the value of ANHC is 0. This packet is then marked by R4 and forwarded via the path R4→R1→R2→R5→R6. Consequently, the packet is caught in a forwarding loop between R5 and R4 until its TTL expires or either link is recovered.

This problem can be trivially corrected by employing an extra bit to indicate a re-routed packet. That is, if a packet encounters a failure, the detecting node also marks it using this bit, in addition to the ANHC. If a marked packet experiences a failure again, it is dropped immediately as the routing technique does not handle multiple failures.

5.6 Suppressing Failure Notification

As has been discussed before, intermittent failures can lead to long episodes of lack of synchronisation between the link-state databases at all nodes, which generates routing instability [52, 73]. Fast re-route using ANHC ameliorates this problem by suppressing the failure notification for a certain amount of time, τ. During this period, destinations affected by link failures can receive packets through ANHC recovery routes with only increased delay, and without jeopardising unaffected routes. As a rule of

thumb, a suppression time τ of around 1 minute for the triggering of a re-convergence process could help to avoid routing instability while maintaining routing flexibility to topology changes. Of course, backup paths and alternate next hops must be re-computed once the routing protocol re-converges.

5.7 Bounds on Alternate Path Length

Intuitively, in case of failure, the resilient routing technique described in Section 5.5 routes packets along the first hops of maximally disjoint paths terminating on their destination node until, after a number of hops equal to the ANHC, a node is reached where they can be routed using the normal shortest path tree of the network. The shortest path route from this node needs not be equal to the backup path calculated by the failure-detecting node - in fact, it is frequently shorter, as it does not need to be maximally disjoint. Thus, the length of the actual path that the packets traverse under failure scenarios is no greater than the length of the backup path to their destination starting from their failure-detecting node.

Furthermore, since all failure-avoiding packets traverse a number of hops equal to their ANHC before they are shortest-path routed, the length of the actual path that the packets traverse under failure scenarios is no shorter than the ANHC to their destination starting from their source.

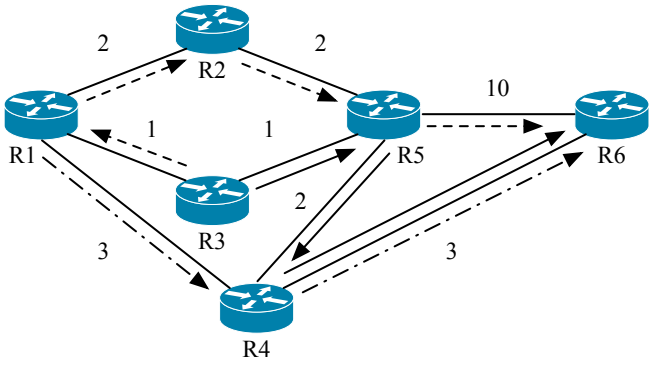


Figure 5.3: An unexpected shorter alternate path from R3 to R6.

Figure 5.3 illustrates an example of an unexpected shorter alternate path from R3 to R6. The normal path of R3 to R6 is R3→R5→R4→R6. Based on Algorithm 5.1, the alternate path known to R3 is R3→R1→R2→R5→R6 and its corresponding ANHC value is 1. When link R3→R5 fails, R3 sets the ANHC to 1, decrements it to 0 and forwards the packet to R1, which is its alternate next hop. The detecting node (*i.e.* R3) expects R1 to forward the packet to R2. Nevertheless, ANHC does not hold a positive number when the packet arrives at R1. Thus, R1 chooses to forward the packet to R4 which is its normal next hop. This causes the packet to traverse along a shorter alternate path that is not known to R3. Although this actual path is unexpected, it does not impact the loop-free property of the routing technique. Moreover, a network operator still has a knowledge on how the traffic is re-directed in the presence of single link failures.

5.8 Optimisation

Evidently, the computations of alternate next hops and their corresponding ANHC values imply additional processing for network elements. Since these only need to be performed for stable topology configurations to pre-compute and cache relevant values (as opposed to be carried out constantly), these can be "amortized" over longer time periods. Thus, it is feasible to perform the algorithm at practical speeds, even using commodity hardware.

If additional efficiency is required, optimised shortest path algorithms can be used. One such algorithm is the incremental Shortest Path First (iSPF) algorithm [75], which avoids the calculation of the whole shortest path tree and instead terminates the computation once the shortest path between a source and a destination is found. This significantly reduces the computation time of the alternate next hops.

5.9 Properties

The two key properties of fast re-route using ANHC are: *a)* full repair coverage for recoverable single link failures and *b)* loop-free forwarding. These properties are guaranteed if the routing scheme is *complete* and *correct* in the presence of *recoverable* failures. For the remainder, it is assumed that equal-cost paths can be distinguished, so that all paths are essentially cost-unique, and all algorithms choose from between equal-cost paths following a deterministic algorithm. Typical ways of achieving this involve differentiating by the number of hops on each path, or choosing on the basis of the interface ID for the first link.

First, Theorem 5.1 shows that fast re-route using ANHC is complete in the presence of any recoverable single link failures.

Theorem 5.1. *If G is not disconnected after the removal of link $(s, n_p(s,d))$, then there exists a path from s to d via $(s, n_s(s,d))$ that does not traverse link $(s, n_p(s,d))$ under fast re-route using ANHC.*

Proof. Let $\mathcal{D}$ be the set of paths from s to d that do not include $(s, n_p(s,d))$. If $\mathcal{D} = \emptyset$, the failure is non-recoverable by definition. Thus, the proof is needed to ensure that when $\mathcal{D} \neq \emptyset$, the ANHC algorithm always finds an alternate path from s to d in $\mathcal{D}$.

The proof is proceeded by contradiction, assuming that $\mathcal{D} \neq \emptyset$ and nonetheless the algorithm knows that $n_p(s,d) = n_s(s,d)$. This implies that the weight of all paths in $\mathcal{D}$ is strictly higher than W_t, since the algorithm adds W_t to the weight of each one of the links of the primary shortest path in order to find the alternate path from s to d, and $w(i,j) > 0 \ \forall (i,j)$. However, the longest path from s to d over G would be an *Eulerian path*, whose weight could be of at most W_t; thus, the weight of all paths in $\mathcal{D}$ could be of at most W_t. Hence, a contradiction is obtained, and $\mathcal{D} \neq \emptyset$ implies that $n_p(s,d) \neq n_s(s,d)$. □

Second, Theorem 5.2 shows that incorporating the pre-computed alternate next hops with the alternate next hop counting mechanism can forward re-routed packets to the destination correctly under failure scenarios. Note that, the packet forwarding in normal case is based on the shortest path tree; hence, it is correct.

Theorem 5.2. *If there exists a path from s to d without link $(s, n_p(s,d))$, fast re-route using ANHC can forward packets from s to d without traversing $(s, n_p(s,d)$.*

Proof. Let $T_p(d)$ be the shortest path tree rooted at d and $H_s(s,d) = \{h_1, h_2, ...\}$ be the hop sequence of the alternate path from s to d excluding s, which is locally known to s. Denote $T_p(d_s)$ as the subgraph of $T_p(d)$ below s, which includes s with a set of vertices N.

Given E is a set of vertices in N that are employed by the alternate path from s to d. Each node e_i in E has the alternate next hop, $n_s(e_i, d)$. As each node e_i shares some links in the $T_p(d)$ with s, $H_s(s,d)$ must involve $n_s(e_i, d)$.

A re-routed packet can encounter a failed link $(s, n_p(s,d))$ if and only if it traverses along $T_p(d)$ starting from any node in E. However, a node forwards a re-routed packet through $T_p(d)$ only if the ANHC value is 0 - after this the packet is no longer routed through alternate next hops.

Since all nodes in E have alternate next hops that coincide with the alternate path from s to d, no re-routed packets arriving at e_i have a zero value ANHC. Thus, packets are not routed along $T_p(d)$ starting from e_i. Furthermore, packets are not routed via $T_p(d)$ starting from a node in N that does not belong to E either, since $N - E$ and $H_s(s,d)$ are disjoint sets.

Finally, routing via $T_p(d)$ from any node outside N does not cause packets to traverse $(s, n_p(s,d))$, because these nodes are not elements of $T_p(d_s)$.

Therefore, it can be concluded that a path for re-routing packets from s to d does not involve the failed link $(s, n_p(s,d))$. □

It is important to note that, alternate path locally known to each router is used for ANHC value calculation only and the routing technique does not hinder the network operator from knowing the actual alternate path used for packet re-routing.

5.10 Performance Evaluation

This section presents the evaluation of fast re-route using ANHC. Similar to Chapter 3–4, the broad areas of investigation include protocol overheads, characteristics of the alternate paths, and the impact of failures and associated recovery mechanisms on link load. The shortest paths upon completion of network re-convergence known as *OSPF re-route paths* are used as benchmarks throughout the evaluation.

5.10.1 Method

Method used in this chapter is similar to that of Chapter 3–4. Self-implemented Java software model is designed to compute alternate next hops and their corresponding ANHC values based on Algorithm 5.1 and Algorithm 5.2. Furthermore, additional extensions are made to determine the network traffic in post-failure scenarios, as the congestion level caused by fast re-route mechanisms should not far exceed those imposed by typical routing re-convergence. In addition to topologies used in Section 4.9, Point-of-Presence (PoP) level topologies are also used. The simulations are run on a machine with a 2.16 GHz Intel Core 2 Duo processor and 2 GB memory.

Unfortunately, real traffic matrices are unavailable for Rocketfuel network topologies. Thus, the gravity model [80] is used to generate traffic matrices composed of edge-to-edge flows. To use realistic

inputs to the gravity model, the data from the U.S. Census Bureau [126] and the United Nations Statistics Division [125] are employed to calculate the city population of each node in the network. However, with Rocketfuel topologies, not all nodes in the backbone area can be mapped onto the actual geographical locations, as the precise location of each node and the corresponding population that is being served by it are unknown. Therefore, PoP-level topologies are used instead for traffic simulation. More importantly, Rocketfuel topologies are not equipped with information on link capacity which is needed for calculating the link utilisation. A technique based on the Breadth-First Search (BFS) algorithm is used to assign link capacity [68]. To examine the traffic characteristics, the traffic matrices are scaled such that the Maximum Link Utilisation (MLU) does not exceed 100% under normal routing or after re-convergence.

5.10.2 Overheads

The followings evaluate different types of overheads:

Computational Overhead

The computation time required for calculating essential parameters to permit fast re-route using ANHC can be divided into: *a)* alternate next hops computation time and *b)* ANHC value computation time. The time complexity for each network topology used in simulations is shown in Table 5.3. Trivially, the times taken to compute alternate next hops and their corresponding ANHC values are directly proportional to the network size. Nevertheless, the largest topology (*i.e.* Sprintlink) requires less than 100 ms for alternate next hops computation[3]. A standard error of each measurement given in Table 5.3 shows that the corresponding result is reproducible. Furthermore, ANHC value computation time is negligible for all topologies. Thus, fast re-route using ANHC does not incur any significant computational overhead and can be employed in normal routers without jeopardising their performance.

Table 5.3: Computational overhead introduced by fast re-route using ANHC.

Topology	**Computation time (ms)**			
	ANHs	**StdErr**	**ANHC**	**StdErr**
Abilene	0.108	0.002	0.006	0.001
GEANT	0.120	0.003	0.008	0.001
Abovenet (AS 6461)	11.920	0.034	0.020	0.001
Sprint (AS 1239)	77.091	0.391	0.051	0.005
Tiscali (AS 3257)	18.981	0.104	0.025	0.003
Waxman	3.091	0.018	0.015	0.002
Barabasi-Albert	3.383	0.015	0.016	0.002
Barabasi-Albert-2	2.554	0.017	0.015	0.002

More precisely, computing the alternate next hops for all node pairs requires $O(V^4 + D \times V^2)$ while computing the number of repetitions requires $O(D \times V^2)$, where V and D represents the number of

[3]The computation time is averaged among all routers.

nodes and the diameter of the network. Therefore, the overall computational overhead can be expressed as $O(V^4 + D \times V^2) + O(D \times V^2)$.

Memory Overhead

In order to enable re-routing with alternate next hop counting, a router must store additional information for each existing destination. That is, apart from the normal next hop, its alternate and the corresponding ANHC value must be maintained. However, no additional routing table entries are required; hence, fast re-route using ANHC does not entail excessive memory overhead.

Packet Overhead

Fast re-route using ANHC requires a few bits in the packet header to store the ANHC value. It is recommended that a part of Type of Service (TOS) field can be used for IPv4, and a part of Traffic Class field can be used for IPv6. Simulation results show that more than 90% of the alternate paths have an ANHC value less than 3. For a large backbone topology such as Sprintlink which has 315 nodes and 1944 links, a maximum ANHC value of 8 is found. In practice, a maximum ANHC value of 7 is needed in the packet, as the failure-detecting node decrements the ANHC value before it forwards the packet to the alternate next hop. Thus, only 3 bits are required for ANHC. To prevent forwarding loops in the presence of multiple failures, an extra bit is used to indicate a re-routed packet. Consequently, fast re-route using ANHC needs no more than 4 bits for an optimal packet re-routing in practical topologies.

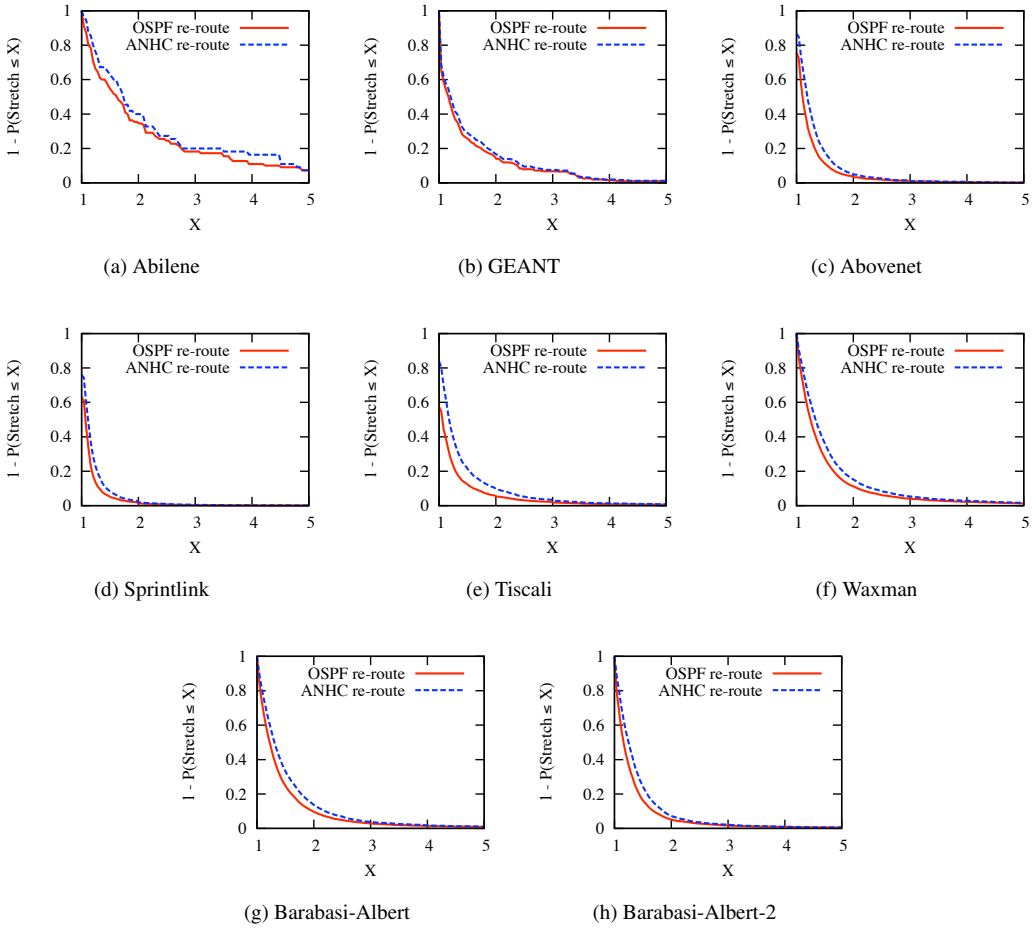
5.10.3 Repair Coverage

As proven in Section 5.9, fast re-route using ANHC guarantees full protection against any recoverable single link failures. Therefore, the repair coverage is always 100%. Comparison with other resilient mechanisms such as LFAs is difficult, since their repair coverage heavily depends on the network topology.

5.10.4 Stretch

It is important to ensure that the stretch is not too high to avoid high delay which is not tolerable in sensitive applications. Figure 5.4 illustrates the stretch of alternate paths across different topologies in comparison with OSPF re-route (*i.e.* the shortest paths after re-convergence). The graphs illustrate inverse cumulative distribution function of the stretch of ANHC in comparison with OSPF re-route. It can be seen that fast re-route using ANHC offers a recovery via alternate paths that are near optimal regardless of the underlying network topology. The average of optimal stretch across all topologies is 1.221 while the average stretch provided by ANHC re-route is 1.305. In most cases, the cost of an alternate path is close to that of the shortest path.

On the other hand, the number of hops of a path should also be considered, as it reflects the number of links a path utilises for routing packets. Figure 5.5 illustrates the number of hops required by the shortest paths, the second shortest paths, and the actual alternate paths calculated based on Algorithm 5.1 and Algorithm 5.2. The average numbers of hops of OSPF re-route paths and fast re-route using ANHC are 4.821 and 4.934 respectively. It can be concluded that fast re-route using ANHC can alleviate the problem of forwarding disruptions due to single link failures using near optimal paths.



(a) Abilene (b) GEANT (c) Abovenet

(d) Sprintlink (e) Tiscali (f) Waxman

(g) Barabasi-Albert (h) Barabasi-Albert-2

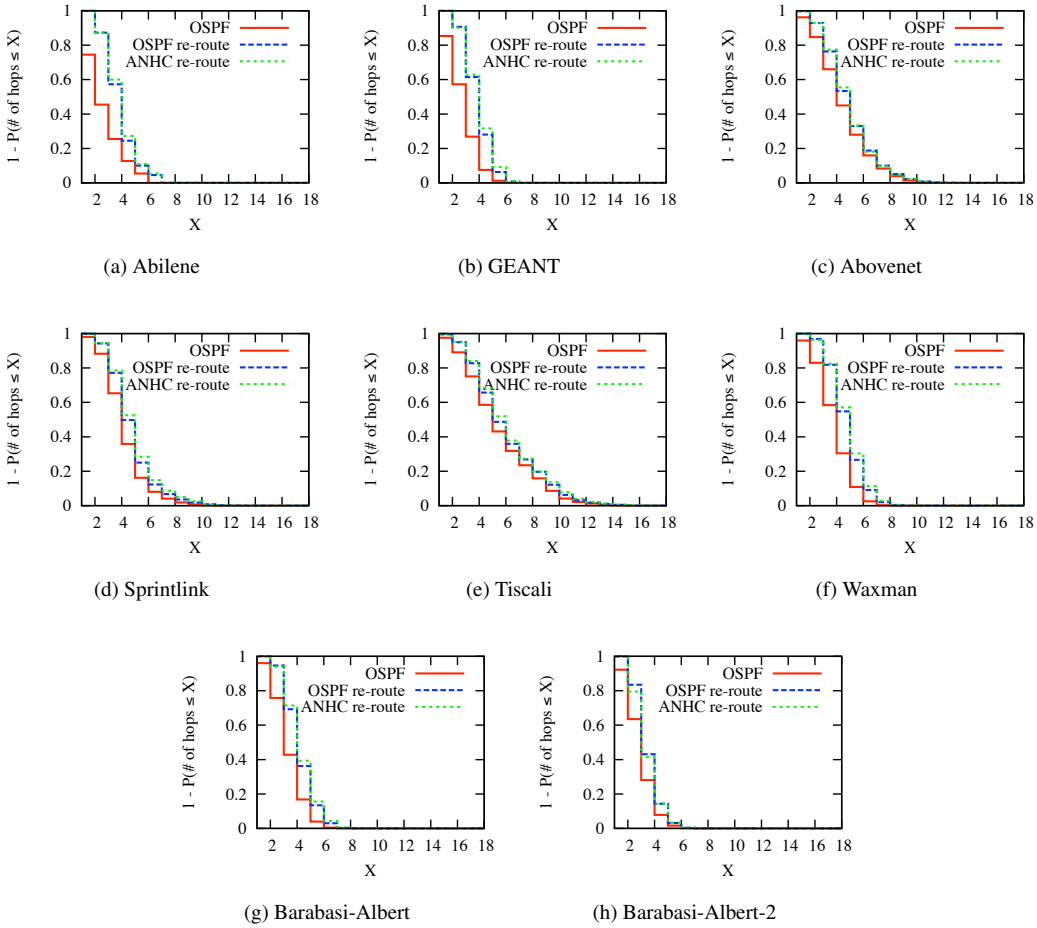
Figure 5.4: Stretch comparison between OSPF re-route and fast re-route using ANHC.

5.10.5 Maximum Link Utilisation

The MLU over different failure scenarios of each topology is used to evaluate the impact of alternate paths on network traffic. Figure 5.6 illustrates the fraction of links in the network with MLU exceeding the value in x-axes. Interestingly, unlike other IPFRR techniques [74], none of the links in all topologies are overloaded under fast re-route using ANHC. Moreover, the technique also performs as well as normal re-convergence in an arbitrary network.

5.10.6 Total Network Overhead

This metric is directly proportional to the number of hops employed by the actual alternate path. It reflects the characteristics of traffic load in the network after failure. The relative increase of the network overhead under fast re-route using ANHC is measured and compared with OSPF re-route for different failure cases. Figure 5.7 shows that there is no significant increase in the traffic load when fast re-route using ANHC is employed. For the Sprintlink network, the worst-case failure causes the network overhead to increase by 7.094% under fast re-route using ANHC while the normal re-convergence increases the traffic load by 6.377%. However, in most cases, the failure incurs an increase in the network overhead less than 2%. The performance difference between OSPF re-route and ANHC re-route is negligible



(a) Abilene (b) GEANT (c) Abovenet

(d) Sprintlink (e) Tiscali (f) Waxman

(g) Barabasi-Albert (h) Barabasi-Albert-2

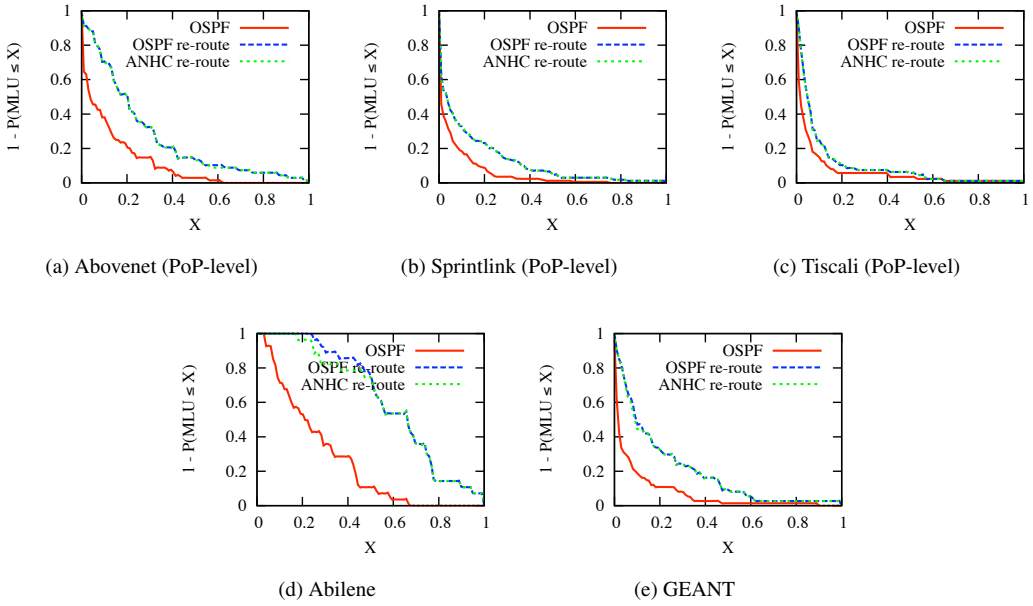
Figure 5.5: Number of hops of a path before and after failures under OSPF re-route and ANHC re-route.

considering the failure occurs only for temporarily.

5.11 Conclusions

A new IPFRR technique based on the Alternate Next Hop Counter (ANHC) used to handle transient link failures was introduced. In the normal scenario, packets are forwarded along the shortest path calculated similarly as in traditional IP routing. When a link fails, the detecting router is responsible for setting the pre-computed ANHC value in the packet header. For each router receiving a re-routed packet, it performs the forwarding based on the ANHC value. If ANHC holds a positive number, the router decrements it by 1 and forwards the packet to its alternate next hop. When the value of ANHC becomes 0, the packet is forwarded along the normal path. Two algorithms for computing the alternate paths and their corresponding ANHC values were presented. Furthermore, the completeness and correctness of the algorithms were proved.

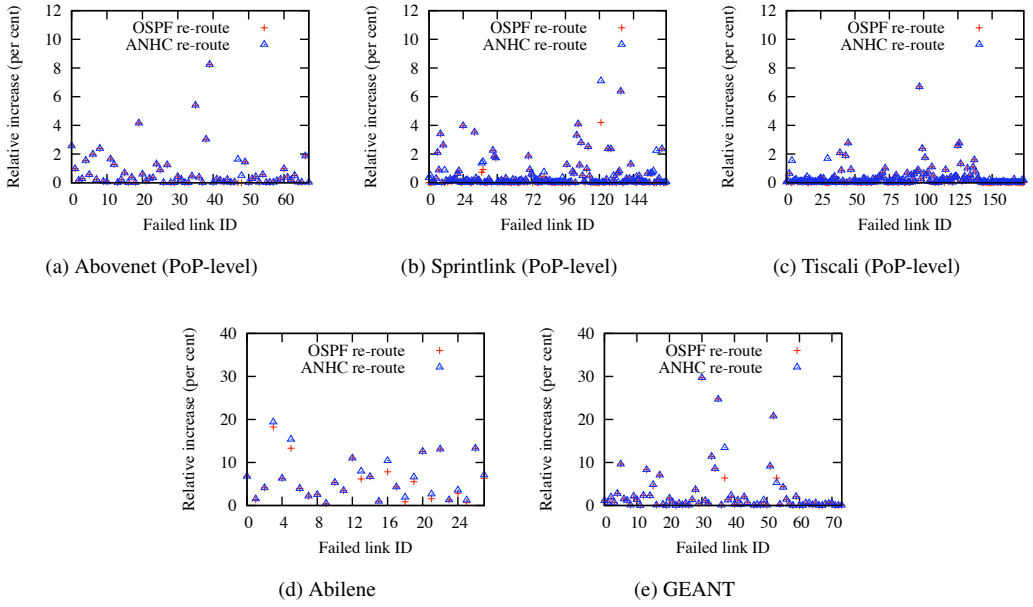
As the paths provided by OSPF re-route are the shortest paths after a failure, they were used as benchmarks throughout the evaluation. From simulation results, it can be concluded that fast re-route using ANHC requires no significant overheads and can be easily deployed without major modifications. Moreover, it can fully protects single link failures using low stretch alternate paths. Both real and syn-



(a) Abovenet (PoP-level) (b) Sprintlink (PoP-level) (c) Tiscali (PoP-level)

(d) Abilene (e) GEANT

Figure 5.6: Maximum link utilisation before and after failures under OSPF re-route and ANHC re-route.



(a) Abovenet (PoP-level) (b) Sprintlink (PoP-level) (c) Tiscali (PoP-level)

(d) Abilene (e) GEANT

Figure 5.7: Relative increase of the total network overhead after failures under OSPF re-route and ANHC re-route.

thetic traffic matrices based on the gravity model were used for examining the impact of fast re-route using ANHC on the traffic load after different failure scenarios. It was illustrated that fast re-route using ANHC does not overload other operable links in the network. It is greatly believed that fast re-route using ANHC can significantly enhance the network reliability without any expensive requirements.

Chapter 6

Resilient Routing Using Packet Re-Cycling

6.1 Introduction

The number of proposed resilient mechanisms in the past few decades has shown that the network reliability is a challenging topic. Nevertheless, most of these approaches aim to increase the resilience by employing backup paths to re-route traffic from single failures, some of which, handle only link failure scenarios. A notable advanced technique that intend to minimise the packet losses from multiple failures, to be precise, any number of failures, is known as Failure-Carrying Packets (FCP) [65].

Although FCP has some interesting properties such as full-protection against network failures and loop-free forwarding, the technique incurs a significant amount of overheads. For example, it requires a large amount of space in the packet header to store the up-to-date failure information. In addition, a real-time computation at each router is also required once an FCP arrives, in order to determine the current available shortest paths to destinations that are affected by the failures.

This chapter introduces a novel approach known as Packet Re-cycling (PR) for routing resilience. PR employs backup paths for fast re-routing packets that encounter failures. Regardless of the concept of backup paths, the definition of these paths in PR is significantly different from those of other resilient routing schemes. Specifically, PR relies on a system of cycles where each uni-directional link in the network is associated with a uni-directional cycle that can be used to bypass itself if a failure occurs, through a process called *cycle-following* described in Section 6.4. This is done in such a way that, if further failures are encountered on links along the backup path, the backup paths of these links are guaranteed to avoid previously encountered failures, for all failure cases where a path still exists between a source and a destination.

Unlike FCP, this is achieved without including failure information in the packet header, or re-calculating routing tables on real-time basis. Each PR-enabled router initialises the protocol by constructing its routing table using a conventional shortest path algorithm (*e.g.* Dijkstra's algorithm). The tables resulting from this process allow forwarding during failure-free conditions, and do not impose any excessive requirements on the current routing paradigm. To equip a router with PR fast re-route capability, each router must implement a *cycle-following table*, an additional information repository that is used to forward packets along the backup paths during the cycle-following process. The additional information required to populate the cycle-following table is obtained from the *cellular embedding* of a

network graph, which is done offline prior to the initialisation of the protocol. Once the embedding is complete, a router can derive the necessary information and construct the cycle-following table.

Once routing and cycle-following tables are constructed, a router can forward packets normally under failure-free cases, while the PR instance triggers a re-routing process when one or more failures are presented in the network. Depending on the location of each re-routed packet, the cycle-following protocol is initialised and terminated along the backup paths. The main features of PR are summarised as follows:

- It offers a full-protection against any number of failures.
- It incurs minimal overheads compared to other resilient approaches.
- It does not require any significant modifications to the traditional routing paradigm, allowing for practical implementation.

The following section describes the cellular graph embeddings, which needs to be done prior to the initialisation of PR.

6.2 Cellular Graph Embeddings

An *embedding* [40] of a graph G on a closed surface S is a way of drawing G on S so that no two distinct nodes coincide and there are no edge crossings. This means that, in an embedding, links become *lines* on S, which only meet at nodes, that become *points* on S. Thus, an embedding of G in S is a way of arranging the nodes and links of G in space so that all of them lie on S and no link crossings occur. To exemplify this, the most basic embedding problem, the embedding of a graph on the sphere (S_0, the orientable surface[1] of genus[2] zero) is reminded. A graph that is embeddable on the sphere is called *planar*, and the issue of graph planarity has been studied in great depth [41, 40, 13]. PR requires this process to be done on an arbitrary surface S, not only on the plane.

This section focuses only on a particular kind of embedding that is specially useful for implementing PR known as the *minimum genus embedding* [86]. This is because it provides a *cellular cycle system*, a system of cycles in G so that every link is included in exactly two cycles.

Thus, just as links have *start* and *end* nodes, in minimum genus embeddings, links have *right* and *left* cycles. Furthermore, a cellular embedding of G on S has the property that the *cells* (the areas over S that are delimited by each cycle in the system) are topologically equivalent (*homeomorphic*) to open discs. Since S is fully covered by them, they also form a *partition* of S (a set of maximal, connected subsets). This means that, after the embedding process is performed, the network can be thought of as a *polyhedron*. In this case, the faces of the polyhedron correspond to the cells, its vertices to the nodes in the network and each polygon delimiting a polyhedral face corresponds to a cellular cycle in G. Formal definitions of these concepts are described below.

[1] A surface is orientable if it does not contain a subspace that is topologically equivalent to Möbius, a space formed from a rectangular strip with a half-twist.

[2] The number of handles on the surface.

Definition 6.1 (Minimum genus embedding)**.** *Minimum genus embedding of a graph G on a surface S has the property that the genus (or non-orientable genus) of S is the minimum possible for the G in question. A system of cycles so that every link belongs to exactly two cycles (i.e. a cellular cycle system) can be used to define a* cellular embedding *of the network on a surface.*

The property of a minimum genus embedding for a graph that makes it ideal for the definition of a cellular cycle system is that it is a cellular embedding [86]. This means that it divides the embedding surface $\mathcal{S}$ in such a way that links are analogous to the edges of a polyhedron, and the regions bounded by them are equivalent to its facets (they are all topologically equivalent to open discs). Thus, a minimum genus embedding allows the network to be understood as a polyhedral structure in which each link is the frontier between two facets, each facet determines a cellular cycle in the network, and the union of all the facets (cycles) in the network spans all its links (twice).

Once a minimum genus embedding is found, the associated facets define a cellular embedding. It is necessary to assign, in a consistent fashion, a rotation sense to each one of the cells. In mathematical terms, this amounts to defining an orientation. Just as a link has a directionality, a cellular cycle has an orientation. These orientations on the embedded G are defined using the right-hand rule [96, 40]. These orientations allow PR to associate, for data transmission over each link and in any particular direction, a *main cycle* that represents the direction of data flow in failure-free conditions and a *complementary cycle*, in the opposite direction, that can be used as a backup if the link has failed.

This allows PR to systematically avoid failures in much the same way that the right-hand rule allows the solution of labyrinths. As the construction of PR is related to the cellular cycle system, which involves graph theory, a network topology is represented by an embedding of its graph consistently throughout this chapter.

6.3 Constructing Routing and Cycle-Following Tables

Once the embedding is complete, each router needs to construct routing and cycling tables. Routing tables are generally obtained as in normal routing protocol such as OSPF and IS-IS. However, the cycle-following table needs to be explicitly constructed. The cycle-following table of a router is a three-column table with i entries, where i is the number of interfaces of a router. Each column stores the information as follows:

- **First column**—indicates the address of the incoming interface of each packet.
- **Second column**—indicates the address of the outgoing interface of each packet corresponding to the incoming interface. This follows the cellular cycling system, which allows PR to cycle-following packets.
- **Third column**—indicates the address of a complementary interface (*i.e.* next hop of the backup path) used to re-route packets if a failure occurs. Basically, each interface stored in this column belongs to the complementary cycle of the corresponding outgoing interface stored in the second column.

Let $I_{Y,X}$ represent an interface at node X receiving packets from node Y. The embedding of a network illustrated in Figure 6.1[3] is used to illustrate the construction of a cycle-following table.

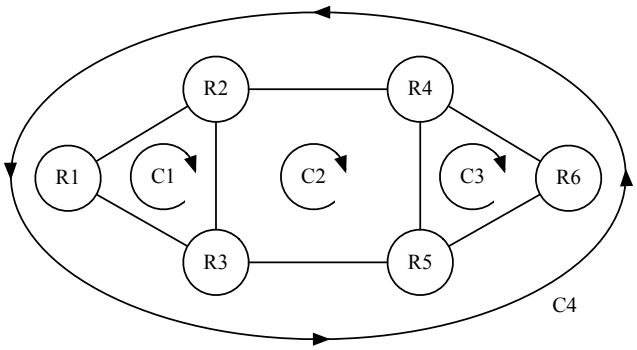


Figure 6.1: A cellular embedding of a simple network topology.

The cycle-following table is constructed based on an oriented embedding of a network graph. Let $I_o(X, Y)$ represent an outgoing interface at node X sending packets to node Y and $I_i(X, Y)$ represent an incoming interface at node X receiving packets from node Y. Using R2 as an example, its cycle-following table is a three-column table with three entries since R2 has three interfaces. Table 6.1 illustrates a cycle-following table at R2.

Table 6.1: Cycle-following table at node R2.

Incoming interface	**Outgoing interface**	**Complementary interface**
$I_i(R2, R1)$	$I_o(R2, R3)$	$I_o(R2, R4)$
$I_i(R2, R3)$	$I_o(R2, R4)$	$I_o(R2, R1)$
$I_i(R2, R4)$	$I_o(R2, R1)$	$I_o(R2, R3)$

Consider the incoming interface $I_i(R2, R3)$, it corresponds in the positive orientation to cycle C2, and in the negative orientation to cycle C1. This means that, if a packet enters R2 through $I_i(R2, R3)$ to follow its positively oriented cycle C2, the packet needs to be forwarded to interface $I_o(R2, R4)$. To determine the complementary outgoing interface, it is noted that cycle C4 is complementary to cycle C2 over the edge implied by the outgoing interface $I_o(R2, R4)$. Thus, in this case the complementary cycle is C4 and the value installed in the complementary outgoing interface of the cycle-following table is $I_o(R2, R1)$, the next hop interface over C4 from R2. In the same way, a packet entering R2 through $I_i(R2, R4)$ should be forwarded using $I_o(R2, R1)$ in order to follow its main cycle, C4, and through $I_o(R2, R3)$ to follow C1, its complementary cycle.

As is clear from the preceding discussion, the first two columns of the cycle-following table can be used to forward packets along the cellular cycles of the network, and essentially are an implementation

[3]It may seem that cycle C4 has the opposite orientation to the other cellular cycles; this is an artifact of the stereographic projection used to represent on the plane what is really an embedding on the sphere. It can be trivially verified that each link belongs to exactly two cycles, each one flowing in opposing direction, thus satisfying the conditions for a cellular embedding.

for the rotation system induced by the cellular embedding [86]. Note that, the forwarding table is a permutation over the output interfaces. In addition, the information stored in cycle-following tables are used to define alternate routes if the corresponding outgoing interface in the second column fails.

6.4 Cycle-Following Protocol

The cycle-following protocol is the key for failure protection in PR. It can be deployed in two different modes: *a)* single failure recovery and *b)* multiple failures recovery. Both modes are operated under different requirements and are described as follows.

6.4.1 Single Failure Recovery

Once routing and cycle-following tables are constructed, PR can guarantee full failure recovery from any single link failures in arbitrary 2-connected networks without any additional information. However, it requires a single bit known as *PR bit* in the packet header. To achieve this, routers forward packets normally according to the routing table, until a failure is detected. At that point, the detecting router marks packets for all affected destinations with the PR bit to indicate that they must be forwarded using cycle-following tables instead of routing tables. After that, these packets are forwarded along the complementary interface associated with the failed outgoing interface. Routers receiving packets with the PR bit set forward them as indicated by the cycle-following interface associated with their ingress interface. By design, each router forwards packets along a cellular cycle (in this particular case, the complementary cycle of the failed link), until they eventually reach the router on the other side of the failed link. When that router attempts to send these packets over the failed link again to continue the cycle-following process, the failure is encountered one more time, and this can be interpreted as a signal that cycle-following is no longer necessary. Therefore, the normal shortest path routing can resume. This signal triggering is also regarded as the *termination condition* of a cycle-following process.

The concept behind this termination condition is as follows. Since all link weights are positive, the next hop is always closer to the destination than the current hop. Thus, once the packet reaches the other side of the failed link, it will not encounter the same failure again if forwarded along the shortest path.

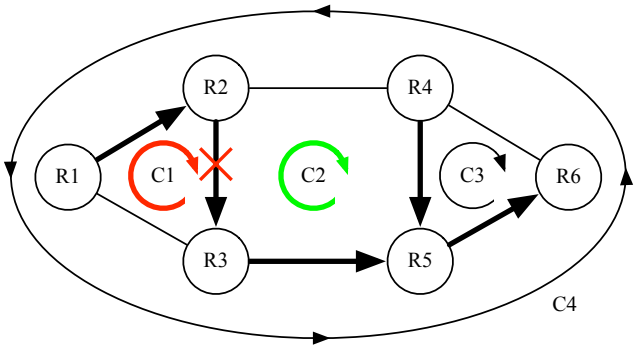


Figure 6.2: Single failure case for illustrating the cycle-following protocol.

Figure 6.2 illustrates an example of a single failure case. Let R1 be the source and R6 be the destination. The bold arrow lines form a shortest path tree rooted at the destination. First, R1 forwards

packets to R2 according to the routing table. Once R2 detects a failure between itself and R3, it realises that the shortest path to the destination is no longer available. Thus, it marks packets destined for R6 and other affected destinations with PR bit and initiates the cycle-following protocol. In this case, only packets originated for R6 are determined. Since the outgoing interface of the failed link corresponds to the positive orientation of C1, R2 forwards packets to R4 through its complementary interface shown in Table 6.1. In addition, other nodes along C2 continue the cycle-following process via the path R2→R4→R5→R3. Once the packets arrive at R3, their PR bit is reset and forwarded to R6 along the conventional shortest path, R3→R5→R6.

It is interesting to note that, even this simple scheme can protect a network from specific instances of multiple link failures. If, for instance, failures are presented not only at R2—R3, but also at R5—R6, the shortest path forwarding would fail at R5→R6 after it resumes the shortest path routing. Nevertheless, the cycle-following process is triggered for the second time at R5 where it detects a failure. From here, the recovery is identical to the previous example.

Algorithm 6.1 summarises the packet processing at each node under PR.

Algorithm 6.1 Packet processing at node s for single failure recovery.

Input: `in_pkt`

```
1:  if in_pkt.PR == 0 then
2:    if Routing.I_out == failed then
3:      in_pkt.PR ← 1
4:      Cycle − follow(in_pkt)
5:    else
6:      Route(in_pkt)
7:    end if
8:  else
9:    if Cycle − following.I_out == failed then
10:     in_pkt.PR ← 0
11:     Route(in_pkt)
12:   else
13:     Cycle − follow(in_pkt)
14:   end if
15: end if
```

6.4.2 Multiple Failures Recovery

Although the aforementioned cycle-following algorithm is capable of handling specific cases of multiple failures, it does not always guarantee a loop-free forwarding. Consider the scenario in Figure 6.3 for example. Let R1 be the source and R6 be the destination. With Algorithm 6.1, packets are forwarded along the normal shortest path until they reach R2 where a failure between R2 and R3 is detected. Consequently, PR bit of the packets is marked and the packets are cycle-followed along the complementary

cycle, C2. However, since another failure is presented, R2 reverts the PR bit to 0 (hence, terminating the cycle-following process) and forwards the packets via the shortest path. As a result, R2 attempts to forward packets via the shortest and cycle-following paths until the TTL expires. In this case, the requirement for packet processing at R2 is increased without gaining any actual throughput. For other cases, where packets are caught in a loop between two or more nodes, the network utilisation also increases.

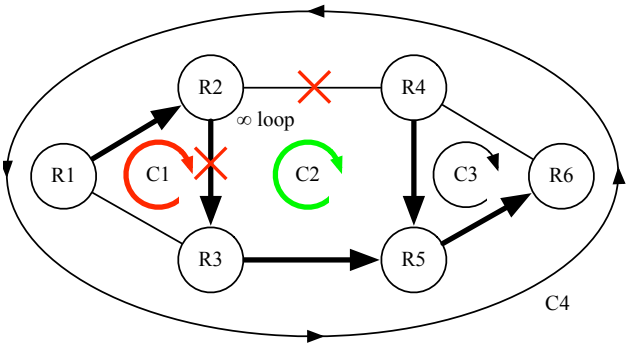


Figure 6.3: Multiple failures case for illustrating the cycle-following protocol.

To solve the problem, PR is enriched by adding an additional column in the routing table that stores the number of hops over the shortest path to each existing destination. The protocol is then updated with a more detailed termination condition known as the *decreasing distance termination condition* or DDT.

When a failure is presented, the detecting router sets the PR bit as in Algorithm 6.1, but in addition, it also marks the packet header with the number of hops remaining along the shortest path to reach the destination from the router behind the link failure. This information is encoded into *CTR bits*. When a packet encounters further failures while cycle-following, each failure-detecting router compares the number of hops from itself to the destination with that one encoded in the CTR bits. If its own distance is smaller, the router overwrites CTR bits with this value and clears the PR bit. After that, the packet can be forwarded along the shortest path. However, if its distance is larger or equal to that of encoded in CTR bits, the packets are forwarded along the complementary cycle of the failed interface.

This mechanism eliminates the aforementioned forwarding loop issue. In addition, it allows PR to handle episodes where many links or nodes[4] are involved. Given the scenario in Figure 6.3, when R2 detects that R2—R3 is down, it sets the PR bit and set 3 as the value of the CTR bits. After that, R2 attempts to send the packets to R4 via C2. However, it also detects that the link R2—R4 is not active. Differently from Algorithm 6.1, R2 compares the value encoded in CTR bits with its own distance to the destination. Since the value stored in the packet header is still smaller, R2 continues the cycle-following process by forwarding the packets along C1 until it reaches the other side of failure at R3. Once R3 receives the packets, it performs similar comparison. However, it realises that the distance from itself to the destination is shorter than that of implied by the CTR bits in the packet header. Thus, it clears the PR bit, replaces the CTR bits with its own distance, and forwards the packets through the shortest path,

[4]A node failure can be regarded as multiple link failures. For example, if a failed node has four interfaces, the scenario is identical to a four link failures scenario.

R3→R5→R6.

Algorithm 6.2 Packet processing at node s for multiple failures recovery.

Input: `in_pkt`

```
1: if in_pkt.PR == 0 then
2:   if Routing.I_out == failed then
3:     in_pkt.PR ← 1
4:     if in_pkt.CTR ≠ 0 then
5:       if Hops(s, in_pkt.d) < in_pkt.CTR then
6:         in_pkt.CTR ← Hops(s, in_pkt.d)
7:       else
8:         in_pkt.CTR ← Hops(s, in_pkt.d)
9:       end if
10:      Cycle − follow(in_pkt)
11:    else
12:      Route(in_pkt)
13:    end if
14:  else
15:    if Cycle − following.I_out == failed then
16:      if Hops(s, in_pkt.d) < in_pkt.CTR then
17:        in_pkt.PR ← 0
18:        in_pkt.CTR ← Hops(s, in_pkt.d)
19:        Route(in_pkt)
20:      else
21:        Cycle − follow(in_pkt)
22:      end if
23:    else
24:      Cycle − follow(in_pkt)
25:    end if
26:  end if
27: end if
```

6.5 Properties

This section proves the correctness of the cycle-following protocol described previously. This is accomplished in three steps. First, the effects of applying the cycle-following protocol with no termination criteria is considered. Second, the proposed termination criteria for multiple failures recovery are proved that they are sufficient to ensure termination of the protocol. Last, the results of the first two steps are used to prove that the cycle-following protocol is guaranteed to deliver packets to their destinations if a

route exists.

6.5.1 Cycle-Following Properties

Consider a network graph that is cellularly embedded on an orientable, closed surface S using a cellular cycle system. As detailed in Section 6.2, the cellular embedding of G guarantees that each edge e either separates two different cells or separates a single cell that is "curved", so that it meets itself along e (in this case, the main cycle and its complement are the same). Of course, this also applies to arbitrary compact regions over S.

It is useful to draw parallels between topological operations involving regions in S (and their boundaries) and the behaviour of the cycle-following protocol. Therefore, it is proposed that a *join* operation is performed on regions in S that share a link along their boundaries. A join performed between two of these regions consists of removing this shared link, taking the resultant connected region as the result of the operation. When a packet encounters a link failure, the path it follows under the guidance of the cycle-following protocol with no termination conditions coincides with a boundary component of the region obtained by joining all cells in S that have at least one failed link on their boundaries. Appendix A[5] provides full proofs for PR. The concept is now used to explore the conditions under which the protocol terminates.

6.5.2 Termination Properties

The cycle-following protocol presented in Section 6.4 induces continuous looping in the network; it is only with explicit termination conditions that this can be prevented. Of course, the conditions proposed in Section 6.4.1 are clearly sufficient for those failure episodes involving a single link. For episodes involving many links, the route that packets take as a result of these failures and whether the termination conditions are sufficient in these cases need to be considered. Appendix A.2 shows that the proposed termination conditions are sufficient. The reason for this is that, when generating new regions by joining cells, the result is always a set of disconnected regions surrounding those nodes and links inaccessible to any given packet source. Then, if a curve following the route over the shortest path tree to the destination encounters one of these regions, it must cross its boundary at least *twice*: once going in, and once going out (otherwise, either the source or the destination would lie within the inaccessible region, implying that there is no available path between them and thus no recovery is possible). By definition, the intersection point going out is closer to the destination than that of the intersection point going in, which is where PR and CTR bits are initially set. Since the packets follow the boundary of the region, the protocol is guaranteed to terminate at the intersection point.

6.5.3 Forwarding Loop Resolution

The argument presented in Section 6.5.2 implies that the termination criteria of Section A.2 are sufficient to ensure that packets do not loop continuously. To this end, it is noted that, the progress of a packet through the network is characterised as a set of intercalated episodes of conventional routing and cycle-following. By definition, the value in CTR bits decreases with each hop when routing. Furthermore,

[5]The properties of PR are formally proved in Appendix A by Raul Landa.

cycle-following episodes always terminate in nodes with lower CTR than that where they started. Thus, since these two kinds of forwarding can only decrease the CTR, which is itself finite (only connected networks with finite weights are considered), a packet is guaranteed to reach the destination in a finite number of steps.

6.6 Performance Evaluation

This section compare the performance of PR in comparison with FCP and normal re-convergence. Due to a limited number of existing mechanisms for full-protection, no other approaches can be used unbiasedly. For consistency, the re-convergence process of routing protocols such as OSPF and IS-IS is regarded as *OSPF re-route* throughout the chapter.

6.6.1 Method

A self-implemented Java software model is designed for analysing the path characteristics of the OSPF re-route, FCP, and PR. These path characteristics include the path length stretch based on link weights and that of based on the number of hops between a source and a destination. Similar to Chapter 3–5, simulations are run on a machine with a 2.16 GHz Intel Core 2 Duo processor and 2 GB memory and the same set of verification methods are used.

In addition, the *planarity* software[6] [16] is used for network graph embedding. The software provides a range of necessary functions for determining the planarity of a network graph as well as extracting the *combinatorial planar embedding*, which provides the adjacency lists in a consistent cyclic order for each vertex. This is very useful, in particular for constructing the cellular cycle system described in Section 6.2.

In this chapter, Abilene [133] and Teleglobe [115] are used for simulations due to its planar characteristic that is required by the planarity software.

6.6.2 Overheads

The followings evaluate different types of overheads:

Computational Overhead

Routing using FCP requires on-demand computation at nodes, while traditional re-convergence requires, in addition, the flooding of failure information throughout the network in order to maintain routing consistency. Although FCP can reduce its computation overhead by requiring routers to maintain per-flow routing state, computation of new routes when an FCP arrives at each router is unavoidable. PR, on the other hand, requires an offline computation of the cellular graph embedding. However, it does not require any additional computational overhead after this initialisation, making PR suitable for real-time operations.

Regarding the complexity of an embedding, the general case is NP-hard [86]. However, linear time algorithms exist for graph embedding on surfaces of known genus [85], which may provide useful 2-cell embeddings for arbitrary networks. In case of planar graphs, very efficient $O(n)$ algorithms are available

[6] Available at: http://code.google.com/p/planarity/wiki/History

[16].

Memory Overhead

The amount of memory that PR requires within each router is incurred by two types of information: *a)* an additional column in the routing table for each existing destination for storing the number of hops to the destination node and *b)* a cycle-following table at each router. Since the former type of memory overhead is commonly required by most resilient mechanisms and the latter one does not incur as much memory requirement, it can be concluded that PR does not impose any significant memory overhead.

Packet Overhead

Regarding the overheads in each packet header, FCP employs more bits in the packet header than are currently available, making its deployment difficult. On the contrary, PR requires a single bit, *PR bit*, to indicate the forwarding mechanism, which protect all single link failure cases while additional *CTR bits* can be used to store the smallest number of hops from the points of failures to a given destination, in order to guarantee a full-protection for any combinations of failures. The size of these CTR bits is in the order of $log_2(d)$, where d is the diameter of a network. To be precise, PR needs $log_2(d+1)+1$ bits in the packet header to ensure that all network links are protected. For example, if a network diameter is equal to 7, the number of bits required is equal to 4. It is recommended that the space in pool 2 DSCP, which is reserved for experimental or local use [94] can be employed.

Note that, OSPF re-route does not incur any memory overhead or impose any requirements for space in the packet header. However, the process cannot be directly compared as its recovery performance is not immediate causing a large amount of packets to drop. This re-convergence is used solely for an analytical comparison.

6.6.3 Repair Coverage

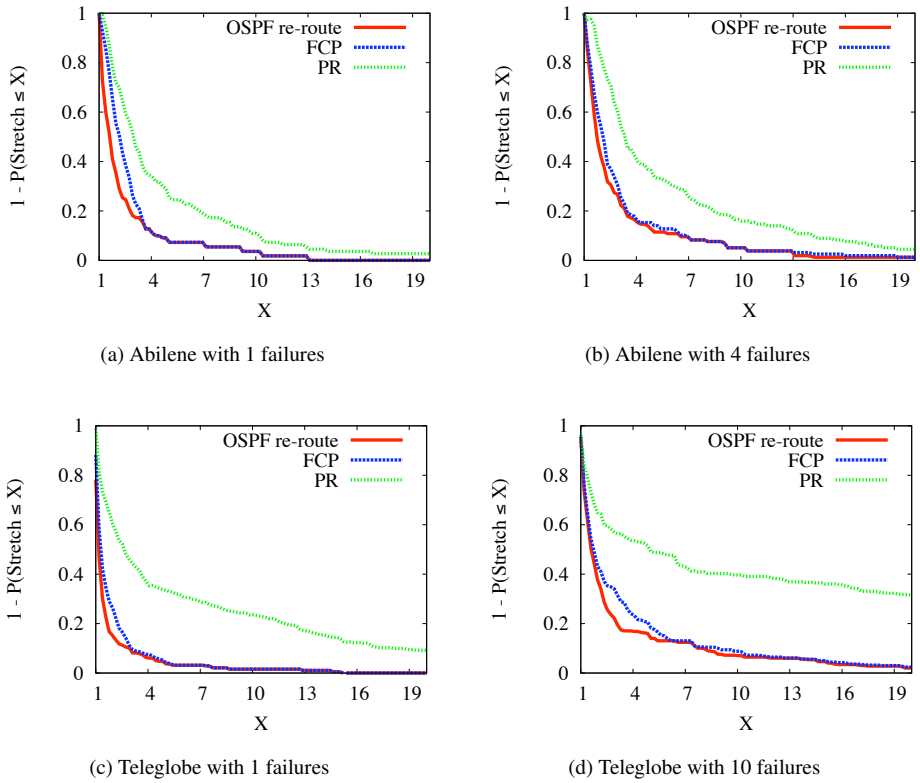
PR properties are proved in Section 6.5. Such properties imply full repair coverage for any number of failures. That means PR always guarantee a repair coverage of 100%. Comparison with other resilient mechanisms is unnecessary as the only known full-protection techniques are FCP and re-convergence.

6.6.4 Stretch

Consistently with prior work, the *stretch* of a path as the ratio between the total path cost while cycle-following and the path cost of the normal shortest path. Figure 6.4 illustrates the stretch of Abilene and Teleglobe under different failure scenarios. The graphs illustrate inverse cumulative distribution function of the stretch of PR in comparison with OSPF re-route.

It can be seen that fast re-route under PR incur considerable amount of stretch compared to other two techniques. This may be the result of the range of weights assigned to links in each network topology. It is interesting to examine the stretch in based on the number of hops instead of the link weights.

Although the link weights are used to calculate the shortest path and the stretch incurred by backup paths as it often reflects the utilisation of a link, manually assigned weights are not necessarily represent the link capacity. Figure 6.5 shows the number of hops incurred by the backup paths computed using normal re-convergence, FCP, and PR. The results in Figure 6.4 and Figure 6.5 are significantly different,



(a) Abilene with 1 failures

(b) Abilene with 4 failures

(c) Teleglobe with 1 failures

(d) Teleglobe with 10 failures

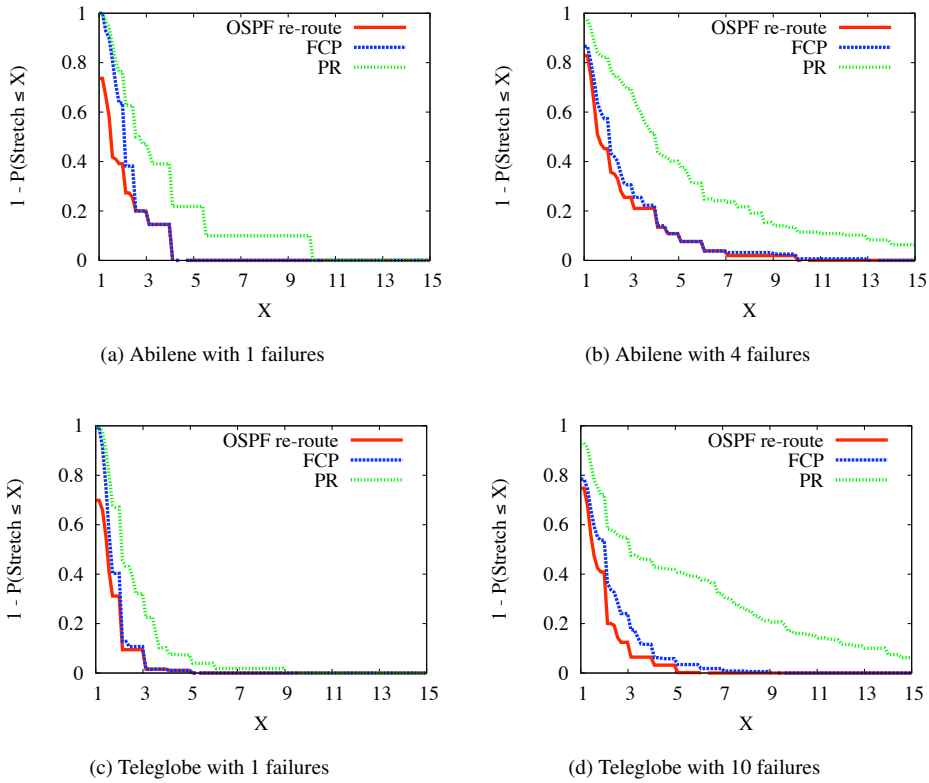
Figure 6.4: Stretch comparison between OSPF re-route, FCP, and fast re-route using PR.

especially for those of Teleglobe. In addition, Abilene with 4 failures cases represent the worst-case scenarios. In other words, if there are 5 or more failures in Abilene network, the network is partitioned into two or more isolated areas.

As PR trades off path length for reliability, path stretch is usually higher than that achieved with FCP. However, since this higher path length stretch is the only price to be paid at forwarding time for full failure protection using very few bits on the packet header, no real-time computations and only very limited memory requirements on routers, PR is still a good alternative to other resilient mechanisms.

PR is an engineering solution that enables network providers and equipment manufacturers to perform several trade-offs. By performing relatively expensive computations off-line, PR releases routers from real-time route re-calculation when failures occur, and by providing an ordered basis for the exploration of backup paths, it allows full failure protection through increased stretch for the saved packets. Overall, PR can be of great usefulness in many IPv4/IPv6 deployment scenarios, particularly because its exceedingly modest requirements in term of packet header space, which might be very useful in cases where such space is restricted or the use of IP options is difficult. Depending on the desired deployment strategy, ISPs can include extra rules and policies to limit PR to certain types of traffic (for example by limiting it to certain classes identifiable by the remaining DSCP bits).

Although PR is designed to offer intra-domain routing resilience, extending this approach to prefixes outside the boundaries of the ISP announced through BGP is possible. Multi-homed ISPs that



(a) Abilene with 1 failures

(b) Abilene with 4 failures

(c) Teleglobe with 1 failures

(d) Teleglobe with 10 failures

Figure 6.5: Stretch based on the number of hops of backup paths under OSPF re-route, FCP, and fast re-route using PR.

receive several announcements for the same prefix via different outgoing links can map this onto a connectivity graph, and use PR to obtain cycle-following routes.

As with all alternate forwarding schemes, PR must cater for the possibility of *link flapping*. This can be done simply by ensuring that link-state transitions only happen after the link remains idle long enough to ensure that packets that encountered the link in its *failed* state do not encounter it again in its *normal* state during the cycle-following process.

6.7 Conclusions

This chapter presented a novel approach for routing resilience known as Packet Re-cycling (PR), which makes use of *cellular cycle system* provided by the embedding of a network graph. Unlike other resilient mechanisms, PR guarantees a full failure protection as long as a source and a destination are not isolated. Advantageously, PR does not impose any additional requirements on the normal routing operations. However, in the presence of failures, PR employs an intercalated episodes of shortest path routing and a mechanism called *cycle-following* to deliver packets to their corresponding destinations for all affected paths. The proofs showed that with PR, a packet always reaches the destination in a finite time regardless of the number of failures presented.

Two techniques, OSPF re-route and FCP were used as benchmarks throughout the evaluation of PR. The simulation results showed that PR incurred considerable amount of path length stretch. However, it

can be argued that the results of re-routing paths created by PR were not optimised and this imposition is the only trade-off PR has in order to protect all failures. In contrast, FCP paths incurred both the requirements for real-time computation and a large space for storing failure information in each packet header. OSPF re-route, on the other hand, does not provide any means for fast re-route; hence, its illustrated performance was not immediate.

Chapter 7

Conclusion and Future Work

This chapter concludes the thesis by summarising the work carried out and suggesting areas for future work.

7.1 Conclusions

Network reliability problem is one of the most concern issues in the current Internet. The overview of routing in the Internet, both inside a single domain and across domains was given in Chapter 2. The chapter also detailed the existing routing protocols as well as identified their strengths and weaknesses. It exposed that the need for convergence process in case of failures and other unpredictable changes can cause a vast amount of data loss. Evidently, this is not tolerable in many applications, especially for those that rely on sensitive data transmission. The lack of resilience has also prohibited many service providers from enabling new range of sophisticated applications.

Existing solutions that has been recently proposed to elevate the problem were described in details. These techniques were analysed in order to identify their benefits and pitfalls. It was concluded that the applicability of routing strategy depends ultimately on the demand of network operators. However, none of the resilient mechanisms can be considered as universal solutions; hence, the research area will still remain challenging for sometime.

Although a convergence-free network can be obtained through the use of Failure-Carrying Packets (FCP) [65], it was mentioned that, shifting out from the current routing paradigm requires a lot of effort and appropriate methods that can offer a seamless migration.

From the analysis, it was clear that none of the existing solutions are optimal. While certain routing strategies can solve the reliability problem in some networks, they may potentially incur side effects such as degradation of router performance. Consequently, a number of alternative fast re-route techniques for routing resilience, which are feasible in practical networks was proposed as follows.

7.1.1 Enhanced Loop-Free Alternates

The Enhanced Loop-Free Alternates, which is also known as E-LFAs was introduced in Chapter 3 to increase the performance of an existing IP Fast Re-Route (IPFRR) solution, Loop-Free Alternates (LFAs) [8]. An evaluation on various properties of the algorithm namely, paths characteristics, protocol overheads, and impacts on network traffic was made to ensure practicability and satisfactory results. Al-

though E-LFAs cannot maximise the repair coverage to a 100% given that network elements are recoverable, they provided near optimal results for all topologies used.

7.1.2 Full Fast Failure Recovery

Graph theories are generally applied in routing to provide efficient methods that construct consistent routing tables. However, some algorithms are both computation and memory expensive. Chapter 4 presented a novel routing technique aiming for failure protection against single link failures known as Full Fast Failure Recovery (F3R). This approach is suitable for small networks without jeopardising other operable parts. F3R relies on link disjoint trees assuming the network is a directed graph. If routing in one tree fails, packets can be routed via another tree successfully if the failed element can be avoided. However, simulation results showed that, although the technique is better than E-LFAs in term of repair coverage (*i.e.* full protection for any recoverable single link failures), it requires significant computational processing time to the extent that it is not acceptable for real deployment. Moreover, the stretch of paths under F3R is very high and can potentially increase the traffic congestion in other parts of a network. Thus, it was concluded that F3R is applicable only for small networks of size less than 100 nodes with low to medium level of utilisation.

7.1.3 Alternate Next Hop Counting

Chapter 5 introduced another IPFRR solution using Alternate Next Hop Counters (ANHC) to provide full protection for any recoverable link failures. The technique uses a forwarding mechanism known as alternate next hop counting where each router receiving a re-routed packet considers its ANHC value and forwards the packet to the next hop accordingly. The technique was evaluated using real, inferred, and synthetic topologies to ensure as accurate and realistic results as possible. In order to ensure that when a failure occurs, the recovery scheme does not jeopardise the rest of the network, the traffic characteristics of post-failure scenarios were also evaluated. The results illustrated that fast re-route using ANHC does not incur any significant overheads or increase the traffic load more than normal routing re-convergence. In addition, it was proved that the technique provides full protection and the pre-computed paths can be used immediately to achieve such performance, while re-convergence process typically requires several seconds to yield similar outcomes.

7.1.4 Packet Re-Cycling

In general, the traditional routing paradigm allows routers to forward packets mainly based on the destination IP address. Moreover, it requires additional resilient mechanisms to alleviate the packet losses during network re-convergence. This prohibits the network from migrating towards the future Internet. Intuitively, a different routing paradigm may be employed as an alternative. Theoretically, an arbitrary network can be decomposed into cycles through cellular graph embeddings. To be precise, a particular kind of embedding known as minimum genus embedding, which provides a cellular cycle system was used to extract these cycles from a network graph. Chapter 6 introduced an alternative approach, Packet Re-cycling (PR), which utilises the cycle system constructed based on the aforementioned embedding. Routing are performed normally when there is no failure presented. However, in the presence of failures,

PR re-routes the traffic using the cycle-following process, which follows the positive orientation of the complementary cycle. PR can be deployed for either single link failure recovery or multiple failures recovery without imposing any expensive requirements.

Simulation results on realistic network topologies showed that, without any optimising technique, PR incurred considerable amount of stretch in the re-routing paths. However, it was considered to be a good trade-off given that it is the only price to be paid for full failure protection.

7.2 Future Work

The future work primarily lies on the improvements and analyses of the PR protocol presented in Chapter 6 as it provides full failure protection. The followings describe the tentative areas of work on PR.

7.2.1 Optimisation of PR

Although it is previously mentioned that PR is capable of handling any number of failures, it is important to note that, the backup paths provided may have high stretch. This is because the technique has not yet been optimised. Some of these longer backup paths may traverse the boundary of a network. Thus, the probability of higher stretch backup paths can be found in larger networks. However, different strategies for overcoming this issue may be used. The future work in this area includes:

Hierarchical Routing

Since PR aims primarily on resilience at an intra-domain level, it is possible to make use of hierarchical routing concept described in a routing protocol such as OSPF. For example, if an OSPF network is divided into a number of areas, each area can be embedded separately. This will fix some of the optimisation problem arisen in PR as follows:

- It reduces the complexity of network embeddings since the size of a graph for each area is smaller than the whole network.
- The stretch can be potentially lowered due to shorter backup paths because the boundary of each area is essentially shorter than or equal to the boundary of its network.
- The number of bits required in the packet header (CTR bits) may be smaller than that of non-hierarchical routing networks due to the fact that the diameter of an OSPF area is equal to (if the number of areas is 1) or shorter than that of the original topology.

However, these benefits have a trade-off as the links connecting areas being unprotected by PR. This issue will also be investigated in the future work.

Dual Network Embeddings

Multi-Topology (MT) routing is one of the approaches that can increase the network reliability. This concept can be used in PR. Chapter 6 described that PR requires a fixed orientation in a cellular cycle system that is provided by the minimum genus embedding. However, this fixed orientation causes high stretch for specific backup paths (under different failure combinations). Implementing two different network embeddings (so-called *dual network embeddings*) with the opposite orientation can elevate this

problem as nodes are capable of choosing the orientation for the cycle-following process so that packets will be forwarded using the embedding with better route.

7.2.2 Analysing the Repair Coverage of Single Bit PR

It is known that with a single bit, PR can handle all single link failure scenarios. However, it can be used to re-route packets from certain multiple failures instances depending on their locations and whether they are recoverable. The future work in this area includes an investigation of the repair coverage of PR for multiple failures using only single bit in the packet header.

7.2.3 NetFPGA Implementation of PR

In order to analyse the applicability and observe the realistic operations of PR, the technique needs to be implemented. Since the deployment in real network is infeasible, NetFPGA[1] will be used to assess the performance issues. In addition, the HEN[2] infrastructure may be used to validate all practical details of the protocols.

The aforementioned future work areas will be investigated thoroughly in order to address vital issues imposed by PR. It is greatly believed that PR will be a very attractive alternative for resilient routing in networks that require high reliability. This will also allow the deployments of services and applications that need reliable data transmission.

[1] A platform used to build a high-speed network switches and routers (http://www.netfpga.org/).

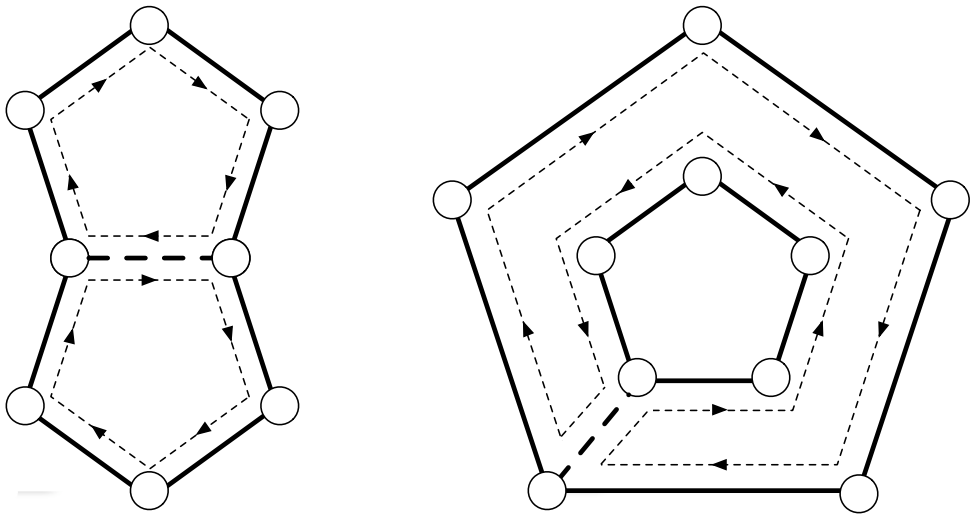
[2] A heterogeneous experimental network operated by the University College Londons Network Research Group (http://hen.cs.ucl.ac.uk/).

Appendix A

Packet Re-cycling Proofs

The following sections prove the properties of Packet Re-cycling (PR) mechanism presented in Chapter 6[1].

A.1 Cycle-Following Properties



(a) Shared edge. (b) Self-contact edge.

Figure A.1: Joins and self-joins of cells.

The proof proceeds from the definitions presented in Section 6.2. Let $G = (V, E)$ be the graph with vertices $V = \{v_1, v_2, ...\}$ and edges $E \in V \times V$ representing the network topology that is cellularly embedded on an orientable closed surface, S using a cellular cycle system, C. As detailed in [86, 40], the cellular embedding of G guarantees that each edge e either separates two different cells, or separates a single cell that is "curved", so that it meets itself along e. In the former case, it implies that the two cells *shared* edge e. In the latter case, edge e is a *self-contact* edge of the cell (see Figure A.1).

Using cell boundaries in G as building blocks, paths between any node pairs can be constructed on a closed surface, S, depending on the set of failures present in the network. It is necessary to define the

[1]The properties in this appendix are proved by Raul Landa.

following terms:

Definition A.1 (Join boundary)**.** *The two distinct cells are considered to have joined along a shared edge, e if the edge is removed; thus it creates a new maximally connected subset of S whose single boundary component is formed by joining the boundaries of the original two cells, once that the shared edge is removed. This creates a closed curve that follows the boundary of the joined cell. This boundary is called the join boundary associated with e in S.*

Definition A.2 (Self-join boundary)**.** *A single cell has self-join along a self-contact edge e if the edge is removed; thus it creates a new maximally connected subset of S whose boundary is the union of the connected boundary components of the original cell, after the removal of the edge. In this case, there are two boundary components, each one associated with one of the nodes that e is incident to. Each one of these boundary components is a closed, continuous curve passing through that node. These boundary components are called the self-join boundary associated with e in S.*

Definition A.3 (Failure profile)**.** *A failure profile, $F \subset E$, is defined as a set of failed edges, unavailable for traffic flow.*

It is advantageous to understand the effects that a given F has on cycle-following over S. Therefore, the definition of failure boundary is defined as follows:

Definition A.4 (Failure boundary)**.** *A failure boundary, ∂F is defined as a boundary induced by the failure profile F in S as the union of all joined boundaries of shared edges in F with all the self-join boundaries of all self-contact edges in F.*

A topological characteristics of ∂F is described using the following theorem:

Theorem A.1. *∂F consists of a set of closed curves. These curves are called F-boundary cycles, which ares denoted as B, so that $\partial F = \{B_1, B_2, ..., B_k\}$ for a failure boundary with k-connected components.*

Proof. Let $F = \{e_1, e_2, ..., e_n\}$ be a failure profile. ∂F can be constructed by performing the joins required by F one by one. Let $\partial F = \emptyset$ and consider e_1, the first edge in F, only two following cases are possible:

- e_1 is a shared edge between two cells. In this case, ∂F becomes the joined boundary of e_1, which is a closed curve.
- e_1 is a self-contact edge of a cell. In this case, ∂F becomes the self-join boundary of e_1, which is a set of two closed curves.

The proof continues with consideration of links $e_i \in F$ where $i > 1$, those e_i which are not yet in ∂F are covered by the previous two cases, and their corresponding closed curves are thus immediately added to ∂F. On the other hand, for those e_i already in F, the following possibilities are considered:

- e_i is shared between ∂F and an external cell. In this case, the cell outside ∂F is joined with ∂F along e and the new border segment acquired in this process becomes homotopic to the original curve between the end points of the failed edge. Therefore, the topological structure of the boundary remains unchanged, remaining a collection of closed curves.

- e_i is shared between two components of ∂F. In this case, when the join is performed, both components are broken; hence, they are removed from ∂F – and a new closed curve formed by joining the two B_j at the two end points of e_i is included in ∂F instead.

- e_i is a self-contact edge of B_j. In this case, when performing the self-join, this curve is broken and removed from ∂F, with the two new boundary components resulting from the self-join being included in ∂F.

□

Theorem A.2 stated below concerns the behaviour of a packet encountering a failure and then being forwarded using the cycle-following protocol described in Section 6.4 with no termination condition.

Theorem A.2. *Let a packet encounter a failed edge, $e_i \in F$ as it is forwarded using conventional routing. If, at that point, the packet is forwarded using cycle-following with no termination condition, the packet follows B_j, the F-boundary cycle induced by F that passes through that point.*

Proof. The proof is proceeded by induction on $|F|$. For $|F| = 1$, let $F_1 = \{e_1\}$ and consider the case where e_1 is a shared link between two cells. In this case, when the failure is encountered, the protocol forwards the packet following the boundary corresponding to the complementary cycle of e_1, which continues until the failure is encountered once more. Then, the packet is forwarded following the boundary of the complementary cycle of the cycle it is currently following; this corresponds to the original cycle which included e_1 in the positive orientation. Thus, for $|F| = 1$, B_1 is just the join boundary of e_1, and $\partial F_1 = \{B_1\}$. Therefore, the only F-boundary cycle of F_1 is followed. In the case of e_1 being a self-contact edge, the packet follows either of the two cycles formed by the failure, depending on what node it starts cycle-following. Since both cycles are F-boundary cycles of F_1 (they coincide with the self-join boundary of e_1), this proves the theorem for $|F| = 1$. Now, assume that the case holds for $|F| = n - 1$, and focus on the behaviour of the protocol with one additional failed link so that $F_n = \{e_1, e_2, ..., e_n\}$ and the failure region becomes ∂F_n. First, note that, if e_n does not lie in B_j^{n-1}, the F-boundary cycle of F_{n-1} that the packet is following by assumption, the followed path remains unchanged, and the theorem is proved. If, however, e_n does lie in B_j^{n-1}, the following scenarios are considered:

- e_n is shared between B_j^{n-1} and an external cell. In this case, it can be seen that the packet follows the complementary cycle of e_n along the border of this cell, until e_n is encountered again, at which point forwarding following the previous cycle resumes. Thus, the route taken coincides with the join boundary of B_j^{n-1} with the cell defined by the complementary cycle of e_n.

- e_n is shared between two components of ∂F. This case is very similar to the previous one. When the packet reaches e_n, they follow B_k^{n-1} the F-boundary cycle with which B_j^{n-1} shares e_n, until the failure is encountered again and forwarding proceeds along B_j^{n-1}. This route coincides with the boundary of the region obtained by joining the two regions which have B_j^{n-1} and B_k^{n-1} as boundaries, which itself is an F-boundary cycle in ∂F_n.

- e_n is a self-contact edge of B_j^{n-1}. In this case, e_n is associated with the same cycle in both the complementary and the positively oriented directions. This cycle must consist, thus, of two cycles starting and ending in the nodes at both ends of e_n, which are then connected through e_n to build a single cycle. Therefore, when the failure at e_n is encountered and the packet is being forwarded along its complementary cycle, the packet simply traverses one of the cycles at the end points of e_n, depending on which end point of e_n is the packet in. Since both of these cycles are in the self-join boundary of e_n, they are both F-boundary cycles in ∂F_n.

□

A.2 Termination Properties

Using Theorem A.2, the termination properties of PR can be proved as follows:

Theorem A.3. *Let $G = \{V, E\}$ be a connected graph of which a cellular embedding is known, and let each $e \in E$ be associated with a cost $c > 0$, which is used to calculate a shortest path tree T rooted at a given destination d. Furthermore, consider a failure profile, $F_n = \{e_1, e_2, ..., e_n\}$ applied to G. Let a packet sent from any source node $s \neq d$ encounters, at a node f, a failed edge $e_f \in F_n$ as it is being routed along T towards d. Then, if there is a path between s and d, the cycle-following protocol described in Section 6.4.1 extended with the termination condition detailed in Section 6.4.2 always terminates.*

Proof. The proof is proceeded by induction on n, the number of failures of the failure profile, F, being considered. For the case $n = 1$ (thus implying $F = \{e_1\}$), the protocol terminates when the failure is encountered for a second time, after the failed link has been avoided and the distance to the destination has been reduced by an amount equal to the cost of e_1 since all link weights are positive. Therefore, the theorem holds for $n = 1$. Assume that the theorem holds for a failure profile F_{n-1} with $n - 1$ failures, and consider the addition of a single extra failure to create $F_n = \{e_1, e_2, ..., e_n\}$. By Theorem A.2, it is known that, before the consideration of a failure at e_n, the packet follows B_j^{n-1}, an F-boundary cycle induced by F_{n-1} that passes through f. If e_n is neither shared with B_j^{n-1}, nor a self-contact edge for B_j^{n-1}, the path followed by the packet is not modified and the protocol terminates at the same node where it terminates when considering F_{n-1}, immediately proving the theorem. Two possibilities are described below:

- e_n is shared with B_j^{n-1}. In this case, consider e_n as being shared either with another F-boundary cycle B_k^{n-1} or with an external cell. In both cases, there is at least one node along B_j^n, the new cycle followed by the packet, where the conditions for termination is known to hold – the node on the other side of e_n. This is because B_j^n must visit all noes that were being visited previously by B_j^{n-1}.

- e_n is a self-contact edge for B_j^{n-1}. In this case, G is disconnected into two disjoint regions, each one having as a boundary component one of the two cycles created by the self-join. Since the case where f is disconnected from node d is explicitly removed, it is known that d lies inside a region

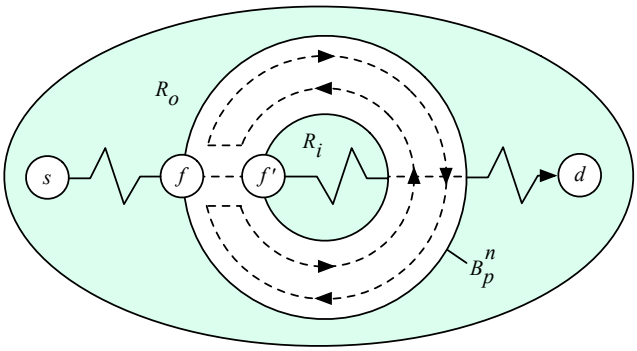


Figure A.2: Cycle-following protocol termination with self-joins.

with an F-boundary cycle that includes f (R_o in Figure A.2). Furthermore, f', the node at the other side of e_n and within the disconnected region (R_i in Figure A.2), is the parent of f in T, and is thus closer to d than f. Since, by construction, there must be a continuous path from s to d under failure-free conditions, this path must start at f, enter the disconnected region through e_n, and then eventually re-enter the connected component where d lies. Thus, there must be a node along B_j^n which is closer to d than f, and the protocol is guaranteed to terminate there.

□

A.3 Forwarding Loop Resolution

As shown in Section 6.4.1, the cycle-following protocol induces continuous looping in the network without explicit termination conditions addressed in Section 6.4.2. The following proves that those conditions proposed in Section 6.4.2 are sufficient to ensure that packets do not loop continuously.

Theorem A.4. *If there is a path available from a given source node s and a given destination node d, the protocol always reaches the destination in a finite time.*

Proof. The proof is simple. First, note that the progress of a packet through the network is characterised as a set of intercalated episodes of conventional routing and cycle-following processes. By definition, the distance towards the destination when routing decreases with each hop. Furthermore, by Theorem A.3, it is proved that cycle-following episodes terminate, and the termination criteria described in Section 6.4.2 ensure that cycle-following episodes always terminate closer to the destination than where they start. Thus, since these two kinds of forwarding can only decrease the distance to the destination, which is itself finite (only connected networks are considered), the destination is eventually reached in a finite number of steps. □

Appendix B

Acronyms and Abbreviations

ABR Area Border Router

Adj-RIB-In Adjacent Information Base, Incoming

Adj-RIB-Out Adjacent Information Base, Outgoing

ANHC Alternate Next Hop Counter

ARPANET Advanced Research Projects Agency Network

AS Autonomous System

ASN AS Number

ATM Asynchronous Transfer Mode

BA Barabasi-Albert

BA-2 Barabasi-Albert-2

BGP Border Gateway Protocol

CCP Cost-Carrying Packets

CIDR Classless Inter-Domain Routing

CLV Code-Length-Value

CSNP Complete Sequence Number PDU

DDT Decreasing Distance Termination

DG-RON Destination-Guided Detouring via Resilient Overlay Network

DIS Designated Intermediate System

DNS Domain Name System

DoS Denial-of-Service

DSC Downstream Condition

DSCP Differentiated Services Code Point

DUAL Diffusing Update Algorithm

DVMRP Distance Vector Multicast Routing Protocol

EBGP External BGP

ECR Edge Controllable Routing

ECMP Equal-Cost Multi-Paths

EGP Exterior Gateway Protocol

EIGRP Enhanced Interior Gateway Routing Protocol

E-LFA Enhanced Loop-Free Alternates

F3R Full Fast Failure Recovery

FCP Failure Carrying Packets

FDDI Fibre Distributed Data Interface

FIB Forwarding Information Base

FIR Failure Insensitive Routing

HMAC - MD5 Hashed Message Authentication Codes - Message Digest 5

IANA Internet Assigned Numbers Authority

IBGP Internal BGP

IETF Internet Engineering Task Force

IGP Interior Gateway Protocol

IGRP Interior Gateway Routing Protocol

IP Internet Protocol

IPFRR IP Fast Re-Route

IPSec IP Security

IPv4 IP version 4

IPv6 IP version 6

IS-IS Intermediate System to Intermediate System

ISP Internet Service Provider

LAN Local Area Network

LFA Loop-Free Alternates

LFC Loop-Free Condition

Loc-RIB Local Routing Information Base

LSA Link-State Advertisement

LSP Link-State Protocol Data Unit

M-ISIS Multi-Topology Routing in IS-IS

MP Merge Point

MT-OSPF Multi-Topology in OSPF

MT-OSPFv3 Multi-Topology in OSPFv3

MIB Management Information Base

MLU Maximum Link Utilisation

MPLS Multi-Protocol Label Switching

MPLS-FRR MPLS Fast Re-Route

MT Multi-Topology

MTU Maximum Transmission Unit

NAT Network Address Translation

NBMA Non-Broadcast Multi-Access

NET Network Entity Title

NGN Next Generation Network

NPC Node-Protection Condition

NS-2 Network Simulator 2

NSSA Not-So-Stubby-Area

oFIB ordered FIB

OSI Open Systems Interconnection

OSPF Open Shortest Path First

OSPFv2 OSPF version 2

OSPFv3 OSPF version 3

PDU Protocol Data Unit

PLR Point of Local Repair

PLSN Path Locking with Safe-Neighbours

PoP Point-of-Presence

PRC Packet Re-cycling

PSNP Partial Sequence Number PDU

QoS Quality of Service

RIB Routing Information Base

RIP Routing Information Protocol

RIP-MTI RIP with Minimal Topology Information

RIPng RIP next generation

RIPv1 RIP version 1

RIPv2 RIP version 2

rLFAs Recursive Loop-Free Alternates

rMRC Relaxed MRC

RON Resilient Overlay Network

RRL Resilient Routing Layers

RSVP Resource ReServation Protocol

RSVP-TE RSVP Traffic Engineering

RT Route Tag

RTE Routing Table Entry

SLA Sevice Level Agreement

SPF Shortest Path First

SRLG Shared Risk Link Group

SRMG Shared Risk Mixed Group

SRNG Shared Risk Node Group

TCP Transmission Control Protocol

TE Traffic Engineering

TLV Type-Length-Value

TOS Type of Service

UDP User Datagram Protocol

VoD Video on Demand

Bibliography

[1] C. Alaettinoglu, V. Jacobson, and H. Yu. Towards milli-second IGP convergence. IETF Internet draft, Nov 2000. http://tools.ietf.org/html/draft-alaettinoglu-isis-convergence-00.

[2] B. Albrightson, J. J. Garcia-Luna-Aceves, and J. Boyle. EIGRP—a fast routing protocol based on distance vectors. In *Proc. Networld Interop*, pages 1–13, Berlin, Germany, Jun 1994.

[3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. The case for Resilient Overlay Network. In *Proc. USENIX HotOS*, pages 152–157, Elmau/Oberbayern, Germany, May 2001.

[4] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Network. In *Proc. ACM SOSP*, pages 131–145, Banff, Canada, Oct 2001.

[5] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda. QoS routing mechanisms and OSPF extensions. RFC 2676, Aug 1999. http://tools.ietf.org/html/rfc2676.

[6] R. Atkinson and M. Fanto. RIPv2 cryptographic authentication. RFC 4822, Feb 2007. http://tools.ietf.org/html/rfc4822.

[7] A. Atlas. U-turn alternates for IP/LDP fast-reroute. IETF Internet draft, Feb 2006. http://tools.ietf.org/html/draft-atlas-ip-local-protect-uturn-03.

[8] A. Atlas and A. Zinin. Basic specification for IP fast reroute Loop-Free Alternates. RFC 5286, Sep 2008. http://tools.ietf.org/html/rfc5286.

[9] R. Bartos and M. Raman. A heuristic approach to service restoration in MPLS networks. In *Proc. IEEE ICC*, pages 117–121, Helsinki, Finland, 2001 Jun.

[10] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[11] D. P. Bertsekas. A simple and fast label correcting algorithm for shortest paths. *Networks*, 23(7):703–709, 1993.

[12] R. Bhandari. Optimal physical diversity algorithms and survivable networks. In *Proc. IEEE ISCC*, pages 433–441, Alexandria, Egypt, Jul 1997.

[13] N. Biggs. *Algebraic Graph Theory*. Cambridge University Press, 2nd edition, 1993.

[14] O. Bonaventure, M. Shand, S. Bryant, and S. Previdi. Loop-free convergence using oFIB. IETF Internet draft, Feb 2008. `http://tools.ietf.org/html/draft-ietf-rtgwg-ordered-fib-02`.

[15] E. Bouillet and J.-F. Labourdette. Distributed computation of shared backup path in mesh optical networks using probabilistic methods. *IEEE/ACM Transactions on Networking*, 12(5):920–930, 2004.

[16] J. M. Boyer and W. J. Myrvold. On the cutting edge: Simplified $O(n)$ planarity by edge addition. *Journal of Graph Algorithms and Applications*, 8(3):241–273, 2004.

[17] S. Bryant, C. Filsfils, S. Previdi, and M. Shand. IP fast reroute using tunnels. IETF Internet draft, Nov 2007. `http://tools.ietf.org/id/http://tools.ietf.org/html/draft-bryant-ipfrr-tunnels-03`.

[18] S. Bryant, M. Shand, and S. Previdi. IP fast reroute using not-via addresses. IETF Internet draft, Feb 2008. `http://tools.ietf.org/html/draft-ietf-rtgwg-ipfrr-notvia-addresses-03`.

[19] R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. RFC 1195, Dec 1990. `http://tools.ietf.org/html/rfc1195`.

[20] B. Chun, J. M. Hellerstein, R. Huebsch, S. R. Jeffery, B. T. Loo, S. Mardanbeigi, T. Roscoe, S. Rhea, S. Shenker, and I. Stoica. Querying at Internet scale. In *Proc. ACM SIGMOD*, pages 935–936, Paris, France, Jun 2004.

[21] T. Cicic, A. F. Hansen, A. Kvalbein, M. Hartmann, R. Martin, and M. Menth. Relaxed Multiple Routing Configurations for IP fast reroute. In *Proc. IEEE/IFIP NOMS*, pages 457–464, Bahia, Brazil, Apr 2008.

[22] J. D. Clercq, D. Ooms, M. Carugi, and F. L. Faucheur. BGP-MPLS IP Virtual Private Networks (VPNs) extension for IPv6 VPN. IETF Internet draft, Sep 2006. `http://tools.ietf.org/html/rfc4659`.

[23] R. Cohen, K. Erez, D. ben Avraham, and S. Havlin. Resilience of the Internet to random breakdowns. *Physical Review Letters*, 85(21):4625–4628, 2000.

[24] R. Coltun, D. Ferguson, J. Moy, and A. Lindem. OSPF for IPv6. RFC 5340, Jul 2008. `http://tools.ietf.org/html/rfc5340`.

[25] S. Deering and R. Hinden. Internet Protocol, version 6 (IPv6) specification. RFC 2460, Dec 1998. `http://tools.ietf.org/html/rfc2460`.

[26] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink. Small forwarding tables for fast routing lookups. In *Proc. ACM SIGCOMM*, pages 3–14, Cannes, France, Sep 1997.

[27] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1(6):269–270, 1959.

[28] M. Domingues, C. Friacas, and P. Veiga. Is global IPv6 deployment on track? *Internet Research*, 17(5):505–518, 2007.

[29] J. C. Doyle, D. L. Alderson, L. Li, S. Low, M. Roughan, S. Shalunov, R. Tanaka, and W. Willinger. The 'robust yet fragile' nature of the Internet. *Proc. of the National Academy of Sciences*, 102(41):14497–14502, 2005.

[30] D. A. Dunn, W. D. Grover, and M. H. MacGregor. Comparison of k-shortest paths and maximum flow routing for network facility restoration. *IEEE JSAC*, 12(1):88–99, 1994.

[31] G. Ellinas, A. G. Hailemariam, and T. E. Stern. Protection cycles in mesh WDM networks. *IEEE JSAC*, 18(10):1924–1937, 2000.

[32] D. Estrin and Y. Rekhter. A unified approach to inter-domain routing. RFC 1322, May 1992. `http://tools.ietf.org/html/rfc1322`.

[33] D. Farinachi. Introduction to enhanced igrp (EIGRP). Cisco's Design Technotes: 13669, Jul 1993.

[34] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[35] P. Francois, O. Bonaventure, M. Shand, S. Bryant, and S. Previdi. Loop-free convergence using oFIB. IETF Internet draft, Feb 2008. `http://tools.ietf.org/html/draft-ietf-rtgwg-ordered-fib-02`.

[36] J. J. Garcia-Luna-Aceves. A unified approach to loop-free routing using distance vector or link states. *ACM SIGCOMM Computer Communication Review*, 19(4):212–223, 1989.

[37] GEANT. The GEANT topology. Online, Dec 2004. `http://www.geant.net/upload/pdf/GEANT_Topology_12-2004.pdf`.

[38] S. Gjessing. Implementation of two resilience mechanisms using multi topology routing and stub routers. In *Proc. IEEE AICT/ICIW*, pages 29–29, Guadeloupe, French Caribbean, Feb 2006.

[39] M. R. Goyal and K. K. W. Feng. Achieving faster failure detection in OSPF networks. In *Proc. IEEE ICC*, pages 296–300, Anchorage, AK, May 2003.

[40] J. Gross and J. Yellen. *Graph Theory and its Applications*. CRC Press, 1999.

[41] J. L. Gross and R. H. Rosen. A linear time planarity algorithm for 2-complexes. *Journal of the ACM*, (4):611–617, 1979.

[42] W. D. Grover. *Mesh-Based Survivable Networks, Options and Strategies for Optical MPLS, SONET, and ATM Networking*. Prentice-Hall, first edition, 2004.

[43] W. D. Grover and D. Stamatelakis. Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for self-planning network restoration. In *Proc. IEEE ICC*, pages 537–543, Atlanta, GA, Jun 1998.

[44] S. H. Gunderson. Global IPv6 statistics—measuring the current state of IPv6 for ordinary users. RIPE 57, Oct 2008.

[45] A. Haider and R. Harris. Recovery techniques in next generation networks. *IEEE Communications Surveys & Tutorials*, 9(3):2–17, 2007.

[46] A. F. Hansen, A. Kvalbein, T. Cicic, S. Gjessing, and O. Lysne. Resilient routing layers for recovery in packet network. In *Proc. IEEE DSN*, pages 238–247, Yokohama, Japan, Jun 2005.

[47] C. Hedrick. Routing Information Protocol. RFC 1058, Jun 1988. `http://tools.ietf.org/html/rfc1058`.

[48] C. L. Hedrick. An introduction to IGRP. Cisco's Technology White Paper: 26825, Aug 1991.

[49] H. Huang and J. Copeland. Hamiltonian cycle protection: A novel approach to mesh WDM optical network protection. In *Proc. IEEE HPSR*, pages 31–35, Dallas, TX, May 2001.

[50] C. Huitema. *Routing in the Internet*. Prentice-Hall, second edition, 2000.

[51] G. Huston. IPv4 exhaustion nears. *The ISP Column*, 2007.

[52] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an IP backbone. In *Proc. ACM IMW*, pages 237–242, Marseille, France, Nov 2002.

[53] A. Iselt, A. Kirstadter, A. Pardigon, and T. Schwabe. Resilient routing using MPLS and ECMP. In *Proc. IEEE HPSR*, pages 345–349, Phoenix, AZ, Apr 2004.

[54] ISO/IEC. Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473). Technical Report 10589:2002, ISO/IEC, Nov 2002.

[55] A. Itai and M. Rodeh. The multi-tree approach to reliability in distributed networks. In *Proc. IEEE SFCS*, pages 137–147, Singer Island, FL, Oct 1984.

[56] A. Itai and M. Rodeh. The multi-tree approach to reliability in distributed networks. *Information and Computation*, 79(1):43–59, 1988.

[57] J. Kang and M. J. Reed. Bandwidth protection in MPLS networks using p-cycle structure. In *Proc. DRCN*, pages 356–362, Alberta, Canada, Oct 2003.

[58] D. Katz, K. Kompella, and D. Yeung. Traffic engineering (TE) Extensions to OSPF Version 2. RFC 3630, Sep 2003. `http://tools.ietf.org/html/rfc3630`.

[59] S. Keshav. *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*. Addison-Wesley, 1997.

[60] S. Kini, S. Ramasubramanian, A. Kvalbein, and A. F. Hansen. Fast recovery from dual link failures in IP networks. In *Proc. IEEE INFOCOM*, pages 1368–1376, Rio de Janeiro, Brazil, Apr 2009.

[61] A. Kvalbein, T. Cicic, and S. Gjessing. Post-failure routing performance with Multiple Routing Configurations. In *Proc. IEEE INFOCOM*, pages 98–106, Anchorage, AK, May 2007.

[62] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne. Fast recovery from link failures using resilient routing layers. In *Proc. IEEE ISCC*, pages 554–560, Cartagena, Spain, Jun 2005.

[63] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne. Fast IP network recovery using Multiple Routing Configurations. In *Proc. IEEE INFOCOM*, pages 23–29, Barcelona, Spain, Apr 2006.

[64] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. *IEEE/ACM Transactions on Networking*, 9(3):293–306, 2001.

[65] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica. Achieving convergence-free routing using Failure-Carrying Packets. In *Proc. ACM SIGCOMM*, pages 241–252, Kyoto, Japan, Aug 2007.

[66] A. Li, P. Francois, and X. Yang. On improving the efficiency and manageability of notvia. In *Proc. ACM CoNEXT*, pages 1–12, New York, NY, Dec 2007.

[67] A. Li, X. Yang, and D. Wetherall. SafeGuard: safe forwarding during route changes. In *Proc. ACM CoNEXT*, pages 301–312, Rome, Italy, Dec 2009.

[68] W. Liu, H. T. Karaoglu, A. Gupta, M. Yuksel, and K. Kar. Edge-to-edge bailout forward contracts for single-domain Internet services. In *Proc. IEEE IWQoS*, pages 259–268, Enschede, The Netherlands, June 2008.

[69] S. Sae Lor, R. Landa, R. Ali, and M. Rio. Handling transient link failures using alternate next hop counters. In *Proc. IFIP Networking, LNCS 6091*, pages 186–197, Chennai, India, May 2010.

[70] G. Malkin. RIP version 2. RFC 2453, Nov 1998. `http://tools.ietf.org/html/rfc2453`.

[71] G. Malkin and F. Baker. RIP version 2 MIB extension. RFC 1724, Nov 1994. `http://tools.ietf.org/html/rfc1724`.

[72] G. Malkin and R. Minnear. RIPng for IPv6. RFC 2080, Jan 1997. `http://tools.ietf.org/html/rfc2080`.

[73] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot. Characterization of failures in an IP backbone network. In *Proc. IEEE INFOCOM*, pages 2307–2317, Hong Kong, Mar 2004.

[74] R. Martin, M. Menth, M. Hartmann, T. Cicic, and A. Kvalbein. The effect of combining loop-free alternates and not-via addresses. Research Report 432, Institute of Computer Science, University of Würzburg, Würzburg, Germany, Sep 2007.

[75] J. M. McQuillan, I. Richer, and E. C. Rosen. The new routing algorithm for the ARPANET. *IEEE Transactions on Communications*, 28(5):711–719, 1980.

[76] M. Medard, S. G. Finn, and R. A. Barry. A novel approach to automatic protection switching using trees. In *Proc. IEEE ICC*, pages 272–276, Québec, Canada, Jun 1997.

[77] M. Medard, S. G. Finn, and R. A. Barry. Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs. *IEEE/ACM Transactions on Networking*, 7(5):641–652, 1999.

[78] D. Medhi and K. Ramasamy. *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann Publishers, 2007.

[79] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: an approach to universal topology generation. In *Proc. IEEE MASCOTS*, pages 346–353, Cincinnati, OH, Aug 2001.

[80] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proc. ACM SIGCOMM*, pages 161–174, Pittsburgh, PA, Aug 2002.

[81] M. Menth and R. Martin. Network resilience through multi-topology routing. Research Report 335, Institute of Computer Science, University of Würzburg, Würzburg, Germany, May 2004.

[82] C. Metz, C. Barth, and C. Filsfils. Beyond MPLS...less is more. *IEEE Internet Computing*, 11(5):72–76, 2007.

[83] G. Meyer. Extensions to RIP to support demand circuits. RFC 1582, Feb 1994. `http://tools.ietf.org/html/rfc1582`.

[84] S. Mirtorabi and A. Roy. Multi-Topology routing in OSPFv3 (MT-OSPFv3). IETF Internet draft, Jul 2007. `http://tools.ietf.org/html/draft-ietf-ospf-mt-ospfv3-03`.

[85] B. Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM Journal on Discrete Mathematics*, 12(1):6–26, 1999.

[86] B. Mohar and C. Thomassen. *Graphs on Surfaces*. The Johns Hopkins University Press, 2001.

[87] D. R. Morrison. PATRICIA—practical algorithm to retrieve information coded in alphanumeric. *Journal of Association for Computing Machinery*, 15(4):514–534, 1968.

[88] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala. Path splicing. In *Proc. ACM SIGCOMM*, pages 27–38, Seattle, WA, Aug 2008.

[89] J. Moy. Multicast extensions to OSPF. RFC 1584, Mar 1994. `http://tools.ietf.org/html/rfc1584`.

[90] J. Moy. OSPF version 2. RFC 2328, Apr 1998. `http://tools.ietf.org/html/rfc2328`.

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

[92] P. Murphy. The OSPF Not-So-Stubby Area NSSA option. RFC 3101, Jan 2003. `http://tools.ietf.org/html/rfc3101`.

[93] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C.-N. Chuah. Fast local rerouting for handling transient link failures. *IEEE/ACM Transactions on Networking*, 15(2):359–372, 2007.

[94] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services filed (DS field) in the IPv4 and IPv6 headers. RFC 2474, Dec 1998. `http://tools.ietf.org/html/rfc2474`.

[95] R. Ogier and N. Shacham. A distributed algorithm for finding shortest pairs of disjoint paths. In *Proc. IEEE INFOCOM*, pages 173–192, Ontario, Canada, Apr 1989.

[96] B. O'Neill. *Elementary Differential Geometry*. Academic Press, 1966.

[97] P. Pan, G. Swallow, and A. Atlas. Fast reroute extensions to RSVP-TE for LSP tunnels. RFC 4090, May 2005. `http://tools.ietf.org/html/rfc4090`.

[98] J. Postel. Internet Protocol. RFC 791, Sep 1981. `http://tools.ietf.org/html/rfc791`.

[99] T. Przygienda, N. Shen, and N. Sheth. M-ISIS: Multi topology (MT) routing in Intermediate System to Intermediate Systems (IS-ISs). RFC 5120, Feb 2008. `http://tools.ietf.org/html/rfc5120`.

[100] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault. Multi-Topology routing in OSPF. RFC 4915, Jun 2007. `http://tools.ietf.org/html/rfc4915`.

[101] T. Pusateri. Distance Vector Multicast Routing Protocol. IETF Internet draft, Oct 2003. `http://tools.ietf.org/html/draft-ietf-idmr-dvmrp-v3-11`.

[102] S. Qazi and T. Moors. Scalable resilient overlay networks using destination-guided detouring. In *Proc. IEEE ICC*, pages 428–434, Glasgow, Scotland, Jun 2007.

[103] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, Jan 2006. `http://tools.ietf.org/html/rfc4271`.

[104] E. Rosen and Y. Rekhter. BGP/MPLS IP Virtual Private Networks (VPNs). IETF Internet draft, Feb 2006. `http://tools.ietf.org/html/rfc4364`.

[105] E. C. Rosen. Exterior Gateway Protocol (EGP). RFC 827, Oct 1982.

[106] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proc. IEEE ICNP*, pages 78–87, Paris, France, Nov 2002.

[107] M. Scheffel, C. Gruber, T. Schwabe, and R. Prinz. Optimal multi-topology routing for IP resilience. *AEU International Journal of Electronics and Communications*, 60(1):35–39, 2006.

[108] A. Schmid and C. Steigner. Avoiding counting to infinity in distance vector routing. *Telecommunication Systems*, 19(3–4):497–514, 2002.

[109] L. J. Seamonson and E. C. Rosen. "STUB" Exterior Gateway Protocol. RFC 888, Jan 1984.

[110] R. Sedgewick. *Algorithms*. Addison-Wesley, 1984.

[111] S. Z. Shaikh. Span-disjoint paths for physical diversity in networks. In *Proc. IEEE ISCC*, pages 127–133, Alexandria, Egypt, Jun 1995.

[112] M. Shand and S. Bryant. A framework for loop-free convergence. IETF Internet draft, Jun 2009. `http://tools.ietf.org/html/draft-ietf-rtgwg-lf-conv-frmwk-05`.

[113] M. Shand and S. Bryant. IP fast reroute framework. IETF Internet draft, Feb 2009. `http://tools.ietf.org/html/draft-ietf-rtgwg-ipfrr-framework-10`.

[114] D. Sidhu, R. Nair, and S. Abdallah. Finding disjoint paths in networks. *ACM SIGCOMM Computer Communication Review*, 21(4):43–51, 1991.

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

[116] A. Sridharan, R. Guerin, and C. Diot. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. *IEEE/ACM Transactions on Networking*, 13(2):234–247, 2005.

[117] D. Stamatelakis and W. D. Grover. IP layer restoration and network planning based on virtual protection cycles. *IEEE JSAC*, 18(10):1938–1949, 2000.

[118] C. Steigner, E. Dickel, and T. Keupen. Rip-mti: A new way to cope with routing loops. In *Proc. ICN*, pages 626–632, Cancun, Mexico, Apr 2008.

[119] J. W. Stewart. *BGP4: Inter-Domain Routing in the Internet*. Addison-Wesley, 1998.

[120] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4(2):125–145, 1974.

[121] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest paths of disjoint paths. *Networks*, 14(2):325–336, 1984.

[122] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker. In search of path diversity in ISP networks. In *Proc. ACM IMC*, pages 313–318, Miami, FL, Oct 2003.

[123] D. Thaler and C. Hopps. Multipath issues in unicast and multicast next-hop selection. RFC 2991, Nov 2000. `http://tools.ietf.org/html/rfc2991`.

[124] D. Torrieri. Algorithms for finding an optimal set of short disjoint paths in a communication network. *IEEE Transactions on Communications*, 40(11):1698–1702, 1992.

[125] United Nations Statistics Division. Demographic and social statistics. Online, Aug 2008. `http://unstats.un.org/unsd/demographic/`.

[126] U.S. Census Bureau. Census 2000 gateway. Online, Apr 2000. `http://www.census.gov/main/www/cen2000.html`.

[127] D. Walton, A. Retana, E. Chen, and J. Scudder. Advertisement of multiple paths in BGP. IETF Internet draft, Jul 2008. `http://tools.ietf.org/html/draft-walton-bgp-add-paths-06`.

[128] B. Xiao, J. Cao, Z. Shao, and E. H.-M. Sha. An efficient algorithm for dynamic shortest path tree update in network routing. *Journal of Communications and Networks*, 9(4):499–510, 2007.

[129] W. Xu and J. Rexford. MIRO: Multi-path Interdomain ROuting. In *Proc. ACM SIGCOMM*, pages 171–182, Pisa, Italy, Sep 2006.

[130] G. Xue, K. Thulasiraman, and L. Chen. Delay reduction in redundant trees for pre-planned protection against single link/node failure in 2-connected graphs. In *Proc. IEEE GLOBECOM*, pages 2691–2695, Taipei, Taiwan, Nov 2002.

[131] X. Yang and D. Wetherall. Source selectable path diversity via routing deflections. In *Proc. ACM SIGCOMM*, pages 159–170, Pisa, Italy, Sep 2006.

[132] M. Zhang, Y. Ruan, and V. Pai. How DNS misnaming distorts Internet topology mapping. In *Proc. USENIX Annual Technical Conference*, pages 369–374, Boston, MA, Jun 2006.

[133] Y. Zhang. The Abilene topology and traffic matrices. Online, Dec 2004. `http://www.cs.utexas.edu/~yzhang/research/AbileneTM/`.

[134] A. Zinin. Analysis and minimization of microloops in link-state routing protocols. IETF Internet draft, Oct 2005. `http://tools.ietf.org/html/draft-ietf-rtgwg-microloop-analysis-01`.