Academic Year 2007 DISSERTATION

Routing Architecture for the Dependable Internet

Yasuhiro Ohara

Graduate School of Media and Governance
Keio University
5322 Endo Fujisawa, Kanagawa, JAPAN 252-8520

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

February 13, 2008

Abstract

Routing technologies are discussed in this dissertation with the goal of improving the network availability. The “highly available Internet” is defined as the Internet with functions to forecast and control the network availability. An implementation strategy to approach to the highly available Internet and to the Dependable Internet is described. This dissertation provides a routing simulator called SimRouting to calculate the network availability, an example of oscillation-tolerant OSPF routing protocol, a family of new multipath route calculation algorithms called MARA, and a new multipath routing architecture called Drouting.

The reliability of communication network depends on various components, for example, the reliability of hardware communication devices, the stability of communication protocols and software systems, the availability of sufficient capacity of network resources, and the reliability of network operators. Current research focuses on a specific one of these reliability components. For example, research on fault-tolerant systems targets the reliability of individual communication devices, circuits, and systems. Research on communication protocols targets their stability. Traffic engineering research focuses on the utilization of the network bandwidth capacity. The current Internet is built upon the combinations of these individual reliability research technologies.

Combining the existing research on reliability of specific components does not improve the practical reliability of the entire communication network for two reasons: 1) Current research does not cover all reliability components. For example, studies on technologies to prevent software bugs, hardware bugs, and mis-configurations by network operators, are not adequate. 2) For many reliability components, no fallback plan is provided when the responsible technologies fail. If a fault-tolerant system or a stabilizing technology for a communication protocol fails, the communication network may cease to function until it is manually repaired. Consequently, the reliability of overall communication network is not sufficient, and the reliability of the Internet is not high enough for mission-critical communication.

Focusing on the routing technologies, this research improves the availability of the entire communication system, as the first step of the reliability research for the Internet. Since the routing technology decides the network path that determines the reliability and the performance of the communication, it is the most important entity for a communication network. Through the aggressive and proactive utilization of multipaths, this dissertation provides a method to bypass trouble spots, even unpredictable ones. This research focuses on the intra-domain routing system.

This research contributes to improving the availability of the IP network through the proposals of 1) a method to estimate the availability of the communication network system considering its routing system, 2) a method to stabilize a routing system, 3) a new routing algorithm to calculate maximum alternative routes in a hop-by-hop network, 4) a new routing architecture to improve the availability of the network via the use of alternative routes, and 5) a method to implement traffic engineering on the new routing architecture. This research as a whole constructs a routing architecture and its peripheral functions that are required for the highly available Internet.

First, this dissertation gives a strategy for improving the availability of the Internet. The highly available Internet is defined to be the Internet with functions required to forecast and control the availability of the network. The highly available Internet is the first and foremost step that must be achieved to approach to the Dependable Internet. Then issues surrounding the availability of the Internet are discussed, focusing on the routing technologies.

This research proposes a method, embodied in a simulation tool, to forecast the availability of communication network systems, taking the routing system into account. A new simulation tool called SimRouting is developed for this purpose. SimRouting can also contribute to the evaluation of routing systems, by providing a way to compare the routing systems in many network configurations. Evaluating routing systems is necessary to improve the performance of the communication network, and hence to improve the reliability of the Internet.

Next, as an example of improving the availability of an individual network element, a technique found in BGP Flap Damping is applied to IPv6 OSPF. It is evaluated in an experimental network and exhibits a significant improvement on the stability.

The Internet was originally constructed to enable utilization of alternative routes. However, there are many problems where one cannot use alternative routes. This research proposes a routing architecture where the network holds multiple routes and the users or the end hosts can utilize one of those multiple routes for recovery from most problems. A family of new multipath route computation algorithms required to implement this architecture, called Maximum Alternative Routing Algorithm (MARA), is developed in order to calculate maximum alternative routes in a hop-by-hop network. Then, a new routing architecture, called the "Drouting architecture", is proposed to employ the MARA routing algorithms. Drouting utilizes a packet tag forwarding method to choose a communication path randomly from many possible routes. Using inferred topologies of real networks, the advantage in terms of failure recovery in the Drouting architecture is shown in simulations.

Traffic engineering is imperative and one of the most important issues in current large scale networks. A traffic engineering method is illustrated to show the feasibility of traffic engineering on the Drouting architecture. A method to preserve both failure recovery property and traffic engineering capability is presented as well.

For almost 30 years the routing system in the Internet has focused on the use of the single shortest path routing. With consideration of feasibility, this dissertation introduces a new concept of using maximum alternative paths. The new routing architecture enables the improvement of the network availability, while allowing network optimization at the same time, without adding any new, complex concepts. The routing architecture is anticipated to be extended further to enable additional feature, such as routing based on the quality of service (QoS). A fundamental routing architecture for the next generation Internet is proposed.

Thesis Committee:

Supervisor:

Prof. Jun Murai, Keio University

Co-Adviser:

Prof. Osamu Nakamura, Keio University
Prof. Takeshi Kawazoe, Keio University
Assoc. Prof. Akira Kato, The University of Tokyo

Acknowledgments

First of all I appreciate my supervisor and co-advisors. My supervisor Prof. Jun Murai directed me for more than ten years, and gave me the opportunity to choose an arbitrary research theme. Without his generous treatment I could not have tackled this research theme in the first place. My co-advisor, Prof. Osamu Nakamura regularly and constantly gave me caustic remarks, hence taught me to have high-targeted perspective and how to grasp and describe the essentials. My co-advisors, Assoc. Prof. Akira Kato and Prof. Takeshi Kawazoe supervised my work in terms of routing technologies and mathematics, respectively.

Mr. Kunihiro Ishiguro provided me with the opportunity to develop a part of a software called GNU Zebra, from which I gained valuable experience on the routing technologies and the implementation. He introduced me to the ways to implement ideas in program codes, how to maintain the quality of a software project, and how to make improvements actually. It was truly an essential for my improvement of my research.

Algorithmic part of this work was produced together with Dr. Shinji Imahori. Discussion and work with him were absolutely necessary to construct the basic technology of this dissertation. Dr. Rodney Van Meter kindly exhibited his enthusiasm for correction of wrong, mistaken, and vague part in technical aspect and even in English expressions.

I also thank to those who gave me precious comments and advices in the Murai Lab.; Assoc. Prof. Hiroyuki Kusumoto, Dr. Noriyuki Shigechika, Dr. Kazunori Sugiura, Dr. Ryuji Wakikawa, Dr. Achmad Husni Thamrin, and Mr. Masaki Minami. I also appreciate those who helped me personally to study both technical and practical things; Mr. Manav Bhatia, Dr. Ken'ichi Nagami, Dr. Hideaki Imaizumi, Mr. Seiji Ariga, Mr. Ayumu Yasuda, and Mr. Shunsuke Fujieda. A number of senior researchers in WIDE Project gave me advices in broader perspective, especially: Dr. Yoichi Shinoda, Dr. Hiroshi Esaki, Dr. Kenjiro Cho, Dr. Kensuke Fukuda, and Dr. Yuji Sekiya. During the days I have tried to receive the degree, I was encouraged much by my colleagues; Dr. Akiko Orita, Ms. Yoko Murakami, Mr. Yusuke Kawakita, and Ms. Shoko Mikawa. I would like to thank also to the administration staffs of Murai Lab.; especially Dr. Keiko Okawa, Ms. Yasue Watanabe, and Ms. Rieko Hori. Colleagues working in the same room with me have kept me from tasks other than the research and have kindly let me concentrate on this work; Mr. Masayoshi Mizutani, Mr. Yohei Kuga, Mr. Yusuke Okumura, Mr. Takeshi Matsuya, Mr. Takaaki Ozaki, Mr. Ryu Sato, and

Mr. Toshiaki Hatano. I appreciate all the members in Murai Lab. for making the place lively and comfortable to work; especially Dr. Keisuke Uehara, Dr. Jin Mitsugi, Dr. Hisakazu Hada, Dr. Jin Nakazawa, Dr. Kenji Saito, Dr. Yasuo Tsuchimoto, Dr. Hideaki Yoshifuji, Dr. Hitoshi Asaeda, Mr. Masafumi Nakane, Mr. Sato Masaaki, Mr. Shigeya Suzuki, Mr. Tatsuya Inaba, Mr. Ryo Kaizaki, Mr. Takaaki Ishida, Mr. Kotaro Kataoka, Mr. Keijirou Ehara, Mr. Tomohiro Ishihara, Mr. Koshiro Mitsuya, Mr. Koji Okada, Mr. Yoshihiko Suko, Mr. Shin Shirahata, Mr. Katsuhiro Horiba, Mr. Tsuyoshi Hisamatsu, Mr. Shinta Sugimoto, Mr. Kazuhiro Mishima, Mr. Noriatsu Kudo, Mr. Yuki Oyabu, Mr. Akira Kanai, and Ms. Yuri Nagai.

My English was greatly supported by the online dictionary web site created by SPACE ALC Inc. I could neither discuss nor write papers in English without the help of the page. I also thank those who gave me peace of mind with music, The Rolling Stones, Led Zeppelin, Yes, and Pink Floyd. The fonts in this dissertation are by the Libertine Open Fonts Project.

Finally I greatly thank my parents and brother, Tomomi Ohara, Rieko Ohara, and Noriaki Ohara, for their patience and supports for long years. Any of my success is also theirs.

Contents

Abstract ii

Acknowledgments v

Contents vii

List of Figures xi

List of Tables xv

1 Introduction 1

1.1 Overview 1

1.2 Contributions of This Work 4

1.3 Interpretation of This Work 5

1.4 Organization of This Dissertation 8

2 Highly Available Internet 9

2.1 Terminologies and Definitions 9

2.2 Illustration of Network Availability 10

2.3 Definition of the Highly Available Internet 11

2.4 Problem Statement 12

2.5 Objective of This Dissertation 13

2.6 Basic Principle of the Approach 14

2.6.1 Multipath Provisioning 14

2.6.2 Multipath Retry 15

2.7 Perspective 16

2.7.1 Network Planning 17

2.7.2 Availability Calculation 18

2.7.3 Maintenance of Network Elements 18

2.7.4 Problem Detection 19

2.7.5 Recovery from Problem 20

2.7.6 Traffic Engineering 20
2.7.7 Service Protection 21
2.8 Strategy 22
2.9 Summary 24

3 The Current Internet Routing 26
3.1 The Internet Topology 26
3.2 Intra-domain Routing Protocols 28
3.2.1 RIP 28
3.2.2 OSPF and IS-IS 29
3.2.3 EIGRP 30
3.3 Inter-domain Routing and BGP 31
3.4 Network Failures 32
3.5 Failure Recovery Methods 33
3.6 Network Administration 33
3.7 Traffic Engineering 34
3.7.1 Routing Metric Optimization 34
3.7.2 Circuit-based Traffic Engineering 35

4 SimRouting: A Tool for Availability Calculation and Routing Evaluation 36
4.1 Overview 36
4.2 Need for Routing Simulation 38
4.3 Requirements for the Simulation Tool 39
4.4 Existing Simulation Tools 40
4.5 Design and Implementation of SimRouting 42
4.5.1 Command Line Invocation and Scenario Files 43
4.5.2 Network Graph Definition 43
4.5.3 Routing Metric Definition 45
4.5.4 Routing Algorithm and Routes 46
4.5.5 Traffic Definition 49
4.5.6 Network State 49
4.6 Example Simulation 50
4.7 Availability Calculation 50
4.8 Evaluation 59
4.8.1 Number of Methods to Construct Network Graphs 59
4.8.2 Achievement of Required Features 59
4.9 Summary 61

5 OSPF Flap Damping **62**
5.1 Overview . . . 62
5.2 Effects of Route Flaps . . . 63
5.2.1 Network Configuration . . . 63
5.2.2 OSPF Behavior . . . 64
5.2.3 Experiment Result . . . 67
5.3 Application of Flap Damping to OSPF . . . 68
5.3.1 Flap Damping Algorithm . . . 68
5.3.2 Experiment Result . . . 69
5.4 Summary . . . 70

6 MARA: Maximum Alternative Routing Algorithm **72**
6.1 Overview . . . 72
6.2 Related Work . . . 75
6.3 Problem Definitions . . . 76
6.3.1 All-to-one Maximum Connectivity Routing Problem . . . 76
6.3.2 All-to-one Max-flow Routing Problem . . . 77
6.3.3 All-to-one Maximum Shortest Path Alternatives Problem . . . 78
6.4 Algorithms . . . 79
6.4.1 MA Ordering . . . 79
6.4.2 MARA-MC . . . 79
6.4.3 MARA-MMMF . . . 82
6.4.4 MARA-SPE . . . 82
6.5 Evaluation . . . 83
6.5.1 Number and Length of Paths . . . 84
6.5.2 Computational Complexity . . . 85
6.5.3 Computation Time . . . 87
6.6 Summary . . . 87

7 Drouting Architecture: A Multipath Routing Architecture **92**
7.1 Overview . . . 92
7.2 Related Work . . . 93
7.3 Deflection Architecture . . . 94
7.4 Drouting Architecture . . . 95
7.4.1 Overview of Drouting Architecture . . . 95
7.4.2 Multipath Route Calculation . . . 97
7.4.3 Tag Forwarding . . . 98
7.4.4 Differences from Deflection . . . 99
7.5 Evaluation . . . 100

7.5.1 Failure Recovery Simulation Method 101
7.5.2 Interpretation of path comparison 101
7.5.3 Simulation and Analysis on Telstra (AS1221) 102
7.6 Observations 108
7.7 Summary 108

8 Traffic Engineering on Drouting Architecture 110
8.1 Overview 110
8.2 Linear Programming Model 111
8.3 An Example of Traffic Engineering 113
8.4 Challenge to Equalize Split Ratio 119
8.5 Summary 123

9 Conclusion 124

10 Future Prospects 126
10.1 Failure Detection 126
10.2 Simultaneous Redundant Packet Transmission 127
10.3 Issues in Traffic Engineering Method 127
10.4 Relation with MPLS Technology 127
10.5 Network Operation and Debugging 128
10.6 Migration and Transition Issues 129
10.7 Inter-domain Routing 130
10.8 Improving the Routing Algorithm 130
10.9 The Distributed Routing Protocol 130

A Theory of Reliability 132
A.1 Systems of components 132
A.2 Coherent system 133
A.3 Paths and cuts 134
A.4 System reliability 134
A.5 Sum of disjoint products approach 136

B Simulations and analysis on other topologies 139

Bibliography 146

Research History 155

List of Figures

1.1 Past Internet. 6
1.2 New Internet. 6
1.3 Interpretation of this work. 7

2.1 Notion of network availability and failure probabilities. 11
2.2 Relations of dependability, Dependable Internet, and highly available Internet. 12
2.3 Multipath provisioning. 15
2.4 Relations of issues and strategy to the highly available Internet. 22

3.1 The hierarchical routing structure of autonomous systems (ASes). 27

4.1 SimRouting architecture. 43
4.2 SimRouting interactive shell. 44
4.3 SimRouting command-line invocation. 44
4.4 SimRouting graph importation. 45
4.5 SimRouting exportation to GraphViz. 46
4.6 SimRouting routing metric specification. 47
4.7 SimRouting route calculations. 48
4.8 SimRouting importation of routing sub-graph. 48
4.9 SimRouting definition of traffic demands. 49
4.10 SimRouting calculation of network state. 50
4.11 SimRouting sample BRITE generated file. 51
4.12 SimRouting sample BRITE generated file (continued). 52
4.13 SimRouting example scenario. 53
4.14 SimRouting example network graph. 54
4.15 SimRouting example traffic demands. 56
4.16 *s*-*t* reliability calculation in the WIDE Project's network. 57
4.17 SimRouting availability calculation scenario. 58

5.1 Network configuration for the route flap experiment. 63

5.2 RTT, OSPF send/receive events during the flapping of 2 seconds cycle (1 second for *Up*, 1 second for *Down*). 65
5.3 The first raised portion of Figure 5.2. 66
5.4 RTT, OSPF send/receive events during the flapping of 7 seconds cycle (*Up* immediately followed by *Down* and remains *Down* for 7 seconds). 67
5.5 RTT, OSPF send/receive events during the flapping of 7 seconds cycle (*Up* immediately followed by *Down* and remains *Down* for 7 seconds) with flap damping. 69
5.6 *penalty* transition during the flapping of 7 seconds cycle (*Up* immediately followed by *Down* and remains *Down* for 7 seconds) with flap damping. . . 70

6.1 Example of routes calculated by SPT and by MARA. 74
6.2 The relation between v, V', v', V'', X and t in Lemma 6.4.1. 81
6.3 The optimality proof of MARA-MC. 82
6.4 Comparison on distributions on the number, the average, and the maximum length of paths for AS1221 and AS1755. 89
6.5 Comparison on distributions on the number, the average, and the maximum length of paths for AS3257 and AS3967. 90
6.6 Computation time of algorithms. 91

7.1 Overview of the Drouting architecture. 97
7.2 A network graph, the shortest path tree (SPT) as it would be calculated today, and the DAG that includes all links. 98
7.3 The Telestra's network topology. 103
7.4 The fraction of failure cases, successfully recovered by different systems on the Telstra topology. 103
7.5 The fraction of 10 failures cases, successfully recovered by different systems on the Telstra topology. 104
7.6 The fraction of source-destination pairs on the Telstra topology, having more than x nexthops. 104
7.7 The fraction of source-destination pairs on the Telstra topology, having more than x paths. 105
7.8 The fraction of source-destination pairs on the Telstra topology, having more than x paths, focused version in the range below 100 paths. 106
7.9 The fraction of source-destination pairs on the Telstra topology, having path length more than x in average. 107
7.10 The fraction of source-destination pairs on the Telstra topology, having path length more than x in maximum. 107

8.1 BRITE configuration file for traffic engineering example. 113
8.2 Network graph structure for traffic engineering example. 114
8.3 Dijkstra's and MARA-MC's routing graph for destination node 0. 114
8.4 Traffic demands for traffic engineering example. 114
8.5 AMPL model file for traffic engineering example. 116
8.6 AMPL model file for traffic engineering example (continue). 117
8.7 AMPL data file for traffic engineering example. 118
8.8 Addition to AMPL model file for equalization of split ratios. 121

A.1 A series structure. 132
A.2 A parallel structure. 133
A.3 An example of reliability computation by minimal path sets. 135

B.1 The Sprint's network topology. 139
B.2 The Ebone's network topology. 140
B.3 The Tiscali's network topology. 140
B.4 The Exodus's network topology. 140
B.5 The Abovenet's network topology. 141
B.6 The fraction of failures, successfully recovered by different systems on the Sprint topology for 1 failure (left) and 10 failures (right). 141
B.7 The fraction of failures, successfully recovered by different systems on the Ebone topology for 1 failure (left) and 10 failures (right). 142
B.8 The fraction of source-destination pairs on the Ebone topology, having more than x nexthops and paths. 142
B.9 The fraction of source-destination pairs on the Ebone topology, having path length more than x in average and in maximum. 142
B.10 The fraction of failures, successfully recovered by different systems on the Tiscali topology for 1 failure (left) and 10 failures (right). 143
B.11 The fraction of source-destination pairs on the Tiscali topology, having more than x nexthops and paths. 143
B.12 The fraction of source-destination pairs on the Tiscali topology, having path length more than x in average and in maximum. 143
B.13 The fraction of failures, successfully recovered by different systems on the Exodus topology for 1 failure (left) and 10 failures (right). 144
B.14 The fraction of source-destination pairs on the Exodus topology, having more than x nexthops and paths. 144
B.15 The fraction of source-destination pairs on the Exodus topology, having path length more than x in average and in maximum. 144

B.16 The fraction of failures, successfully recovered by different systems on the Abovenet topology for 1 failure (left) and 10 failures (right). 145
B.17 The fraction of source-destination pairs on the Abovenet topology, having more than x nexthops and paths. 145
B.18 The fraction of source-destination pairs on the Abovenet topology, having path length more than x in average and in maximum. 145

List of Tables

3.1 Growth in number of ASes and IP prefixes. 27

3.2 Number of routers and links in Splintlink, Verio, and AT&T. “BB” refers to backbone. 28

4.1 Results of example simulation in SimRouting. 55

4.2 Results of the *s*-*t* reliability calculation in SimRouting. 57

4.3 Supported methods in SimRouting. 59

4.4 Comparison of features with other simulation tools. 60

6.1 Average and standard deviation of the number, the average length, and the maximum length of paths. The average length increases with multipath algorithms due to the active use of secondary, longer paths. 84

6.2 Computational complexities of routing algorithms. l denotes the number of neighboring nodes for a node, m the number of edges in the graph, and n the number of nodes. 85

7.1 The network graphs for simulations of Drouting architecture. 100

8.1 Link utilizations routed by Dijkstra. 115

8.2 Link utilizations routed by MARA-MC. 115

8.3 Link utilizations comparison after solving LP. 119

8.4 Split ratio calculated by LP. 120

8.5 Link utilizations comparison of modified LP problem. 122

8.6 Split ratio calculated by modified LP problem. 122

Chapter 1

Introduction

1.1 Overview

The Internet is a packet switched communication network constructed of many sub-networks, which connect a huge variety of devices and media. The communication protocol for the Internet, Internet Protocol (IP) [77], is specified so that it can exchange data on any datalink technologies. Thus the Internet utilizes an almost unlimited array of types of media such as coaxial cables, twisted-pair copper cables, optical fiber cables, and many types of radio transmissions including satellite. The Internet is incorporating and replacing many existing networks, such as Public Switched Telephone Network (PSTN) and television broadcasting network as well as many originally-service-dedicated data networks.

The Internet has become huge and is still growing explosively. ISC Internet Domain Survey [43] estimated an Internet host count; in January 1995 it was $5,846,000$ (adjusted statistics), and had grown to $433,193,199$ by January 2007. According to CIDR Report [38], the number of autonomously administrated networks (the number of ASes) was approximately $25,000$ and the number of IP sub-networks (i.e., routes) was approximately $230,000$ in June 2007. According to the estimation of Internet World Stats [44], there were 719 million users on the Internet in 2004 and 1093 million in 2006.

The Internet has become an essential communication infrastructure for daily life. Many human activities conducted by various people for various purposes now use the Internet. The list of applications that use the Internet includes: communications, media content streaming, commerce, and health and safety. All these applications indicate the importance of the reliability of the Internet.

Communication applications include e-mail, chat, instant message, blog, bulletin board, commercial advertisement, newspaper, magazine, and press release. Communications are used by people very frequently, and hence they are desired to be performable without any stress. Streaming applications, which include IP phone, live television broadcast, and watch-ing movies require a significant amount of bandwidth and stability on packet jitter, and some of them prefer small communication delay. In commerce applications including Internet shopping, exchange of money, and online trades of stocks, we handle our money and property, and hence loss or alteration of transaction data is not permissible. Health and safety applications such as remote medical care, telesurgery, and emergency phone calls, relate to human health and life, and thus require no loss, no errors, and very small delays in communication, and may require significant amounts of bandwidth as well as privacy and security.

As an infrastructure that supports these diverse applications, and because some of them are very important for our life, the reliability of the Internet is really important. However, the reliability of the Internet is still insufficient in order to serve as an integrated communication infrastructure that can support mission critical services. Communication in the Internet can be, and actually is, disrupted for various reasons.

First, routers and links are made down due to network faults such as optical fiber cuts and router failures, and network maintenance such as router software upgrades. Past studies showed that such faults could last an hour or two [50, 49, 40, 39, 56]. Second, router software bugs can result in the complete meltdown of the entire network, as seen in the AT&T frame relay outage in 1998 [64]. Third, even if it is carefully designed, a distributed algorithm may have a defect that arises only on rare incidents such as bit-drops in a router's memory. Such defect can result in the complete meltdown of the entire network, as seen in the ARPANET in 1980 [84]. Fourth, a mistake in network design or network administration can halt the entire network, such as seen in NTT-East in Japan in 2007 [26, 66]. Last, the network connection may be manually shutdown, for example due to a disagreement on the connection between two Internet Service Providers (ISPs), as seen in U.S. in 2005 [20]. In such an event, some customers of the ISPs suffered from reachability problems for particular destinations.

The Internet is going to be used for various applications including mission critical services such as emergency phone calls. Although emergency phone calls are the most important communication that we desire to be reliable, the current Internet is not reliable enough to support them. There is no assurance, estimation and/or expectation for a emergency phone call to be successfully established and continuable for a desired duration. While there are other requirements to implement emergency calls such as call traceback, first and foremost the reliability of the Internet must be improved.

The reliability of a communication network is affected by various factors, such as the reliability of hardware communication devices, the stability of communication protocols and software systems, the availability of sufficient capacity of network resources, and the reliability of network operators. Most of the current research focuses only on a specific one of these reliability components. For example, research on fault tolerant systems targets the reliability of individual communication devices, circuits, and software systems. Research on communication protocols targets on their stability. Traffic engineering research focuses only on the usage of network bandwidth capacity. Today's Internet is built upon the combinations of these research technologies that each targets its specific reliability.

Combining the existing research technologies for a specific reliability component does not improve the practical reliability of the entire communication network. This is because of two reasons: 1) Current research does not cover all reliability components. For example, studies on technologies to prevent software bugs, hardware bugs, and mis-configurations by network operators are not adequate. 2) For many reliability components, no fallback plan is provided when the responsible technologies fail. If a fault tolerant mechanism or a stabilizing technology for a communication protocol fails, the communication network may cease to function until it is manually repaired. Consequently the reliability of overall communication network is not sufficient, and the Internet cannot yet serve as a safety-critical infrastructure.

As the first step of approach to the improved reliability of the Internet, this work focuses on the *availability* of the network. The availability is always required, and the most important attribute of the Dependable Internet. Improvements in the network availability certainly contributes to approaching the Dependable Internet. This dissertation provides, as its results, the minimal routing technologies to forecast and control the network availability in the Internet, i.e., the routing technologies required in the highly available Internet.

Definitions of the Dependable Internet, the highly available Internet, and the network availability, the relations between them, a strategy to implement the highly available Internet, and a strategy to approach to the Dependable Internet are described in Chapter 2. This work proposes realistic steps toward the highly available Internet. The strategy, the scope of this work, and the forecasts for the other areas that are outside the scope of this work are given.

This work focuses on the routing technology. When considering the network availability, the routing technology is the most important topic, because the routing system decides which devices and circuits the communication traverses, hence it also decides how the success probability (i.e., the network availability) of the communication will be. This thesis focuses on intra-domain routing system as the first step.

A method to calculate the network availability, a method to stabilize a routing protocol, and a novel routing architecture are proposed in this work in order to contribute to the improvement of the availability of the overall communication network system. A routing simulation tool, SimRouting, has been developed to provide a method to calculate the network availability on various network configurations. SimRouting contributes to the comparative evaluation of new routing systems, and thus is expected to contribute to improving the routing systems. Then, a method to stabilize a routing system is presented, as an example of increasing the robustness of individual network elements. Next, a novel multipath routing architecture, Drouting, is proposed. The Drouting architecture consists of three parts: routing algorithms that calculate multipath routes, a packet tag forwarding system, and the traffic engineering on it.

Focusing on the routing technology, this research improves the availability of entire communication system, through the strategy of utilizing an alternative route in the face of troubles caused by various sources.

This work proposes basic functions that are required to realize the highly available Internet. The basic functions provide practical solution to the reliability problem. With considerations to the future issues, this work proposes a basic strategy to improve the availability of the Internet.

1.2 Contributions of This Work

The contributions of this work are as follows.

1. The development of a new simulation tool to calculate network availability and to evaluate routing systems is described.

   The simulation tool, SimRouting, can be used to calculate the network availability, to predict which events can affect availability by causing, and to help determine where investment in new infrastructure is most needed. Hence, SimRouting is anticipated to help network planning. SimRouting also can evaluate a routing method on various network graphs, traffic demands, and settings of the routing metrics. The ability to verify a routing system on various network configurations and the ability to compare two different routing systems on the same network configuration help to evaluate and improve the routing systems. Hence, the tool can also be used by routing researchers to compare and evaluate proposed routing systems with existing ones, to improve them. The SimRouting routing simulation tool was used to evaluate the new routing system proposed later in this work.

2. A method to stabilize a routing system is shown.

   A technique found in BGP Flap Damping [103] is utilized to stabilize an OSPF [59] routing system. Because the same kind of approach can be applied to other distributed software system, this work demonstrates a method to make distributed software systems tolerant to many kinds of oscillation problems.

3. A new multipath routing algorithm is proposed.

   New hop-by-hop network routing problems, the all-to-one maximum connectivity routing problem, the all-to-one max-flow routing problem, and the all-to-one maximum shortest path alternatives problem, are defined, and the optimal algorithms to solve these problems, MARAs, are presented. These problems and the optimal algorithms are new findings in Graph theory and network flows. MARAs can be used as a general multipath route calculation algorithm, to compute many multipaths on hop-by-hop networks.

4. A new multipath routing architecture is proposed.

   A new multipath routing architecture, called Drouting architecture, is developed to construct a highly available communication network. Although the focus in this work is limited to the application to the IP network, the same logic can be applied to other dedicated communication networks and/or control networks. In Drouting architecture, a packet tag corresponds to a network path. Changing the packet tag instructs a stochastic change the network path, although it is unknown which path the result would be. This enables the immediate failure recovery when an application in the end host or an intermediate router detect a failure, without waiting the routing convergence. Since just changing the packet tag instructs the network path change, many network entity can utilize the feature to switch to another path. This further enables the implementation of network path evaluation function on many network entity, depending on the specific purpose of the application or the network entity.

5. The feasibility of traffic engineering on the new routing architecture is shown.

   Traffic engineering on the Drouting architecture is examined to show that both failure recovery and network optimization can be performed. It indicates the new vision of the future Internet where the Drouting architecture is applied.

1.3 Interpretation of This Work

Figures 1.1 and 1.2 presents the before and after this work, respectively, regarding the concept of features of failure detection and route change in IP routing. In the past (Figure 1.1), failure detection and route change are performed internal to the each IP routing protocol. Route change function is only triggered by its own failure detection function. In the new Internet after this work (Figure 1.2), route change function is open to other network entities, with the application interface which is to change the packet tag. Since many other network entities can utilize route change functions in intermediate IP routers, other entities such as applications and transport protocols can implement its own failure detection for their specific purpose, and can instruct intermediate IP routing systems to change the communication path. Although the route change is limited only to just "change" and there is no way to specify a specific network path, this is recognized as a partial achievement of "user control".

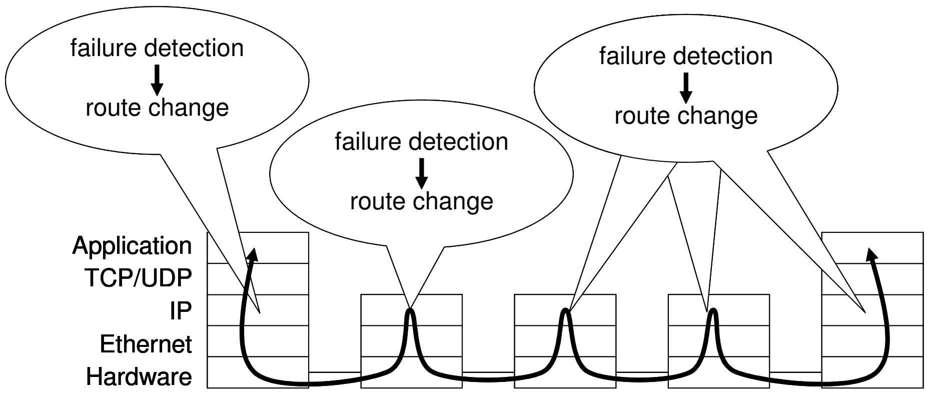

Figure 1.1: Past Internet.

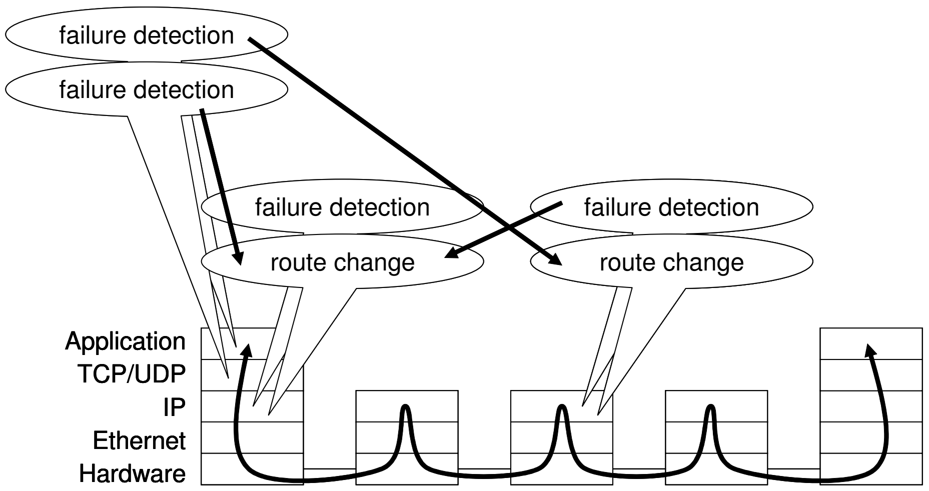

Figure 1.2: New Internet.

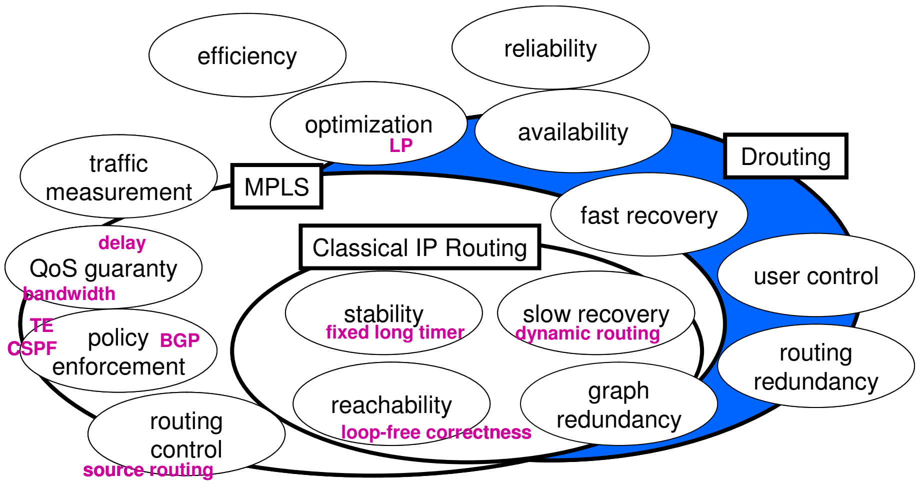

Figure 1.3: Interpretation of this work.

Figure 1.3 illustrates the interpretation of the contribution of this work.

Classical IP routing systems had following properties. The "`stability`" was preserved by the fixed long timer in the routing protocols, along the principle of preferring stability over the adaptability. This led to the "`slow recovery`" nature of the dynamic routing protocols. Providing the stable "`reachability`" was achieved, as the primary purpose of the routing systems. The "`graph redundancy`" was achieved by network planning activities. The "`policy enforcement`" was partially achieved by Border Gateway Protocol (BGP) [82].

Multi-Protocol Label Switching (MPLS) [83] added some features to the Internet. The "`traffic measurement`" can be performed easily, when many Label Switched Paths (LSPs) are established and the LSP statistics are recorded. The LSP enables "`QoS guaranty`" such as guaranteeing delay and bandwidth requirements to a application, "`policy enforcement`" such as traffic engineering (TE) and constrained shortest-path first (CSPF) calculation, and detailed "`routing control`" by the source routing.

Drouting architecture developed in this dissertation adds further features to the Internet. MARAs used in Drouting architecture achieve "`routing redundancy`" by multipaths, in contrast to the single shortest-path routing performed before. Since changing the packet tag can be performed even in the user application, "`user control`" is achieved partially (it is not detailed control). Network path can be switched by the packet tag change, hence "`fast recovery`" can be achieved where one can change the network path immediately after detecting the failure. This fast recovery achieves the improvement of "`availability`", and approaches to "`reliability`". Additionally, the Drouting architecture introduce the easy way to perform "`optimization`" of the network, by using the Linear Programming

(LP) for traffic engineering. This improves "`efficiency`" of the Internet.

1.4 Organization of This Dissertation

The rest of this dissertation is organized as follows. Chapter 2 defines the highly available Internet as an objective of this research, and then provides the required functions, a strategy toward the highly available Internet, and the scope of this research. For the issues outside the scope of this research, related works and forecasts from the perspective of this research are also given. Chapter 3 gives the current Internet routing technologies and status of real network operations as the backgrounds of this research. In Chapter 4, the routing simulation tool, SimRouting, is developed. The example usages are described, and the SimRouting is evaluated by comparisons of functionalities with other network simulation tools. Chapter 5 studies to stabilize the OSPF, as an example to improve the availability of an individual network element. It is shown that the OSPF routing system can be made more robust in the face of an oscillating route, by using a flap dampening feature that is originally introduced to BGP [82]. Chapter 6 exhibits new multipath routing algorithms, MARAs, that maximize the minimum connectivity and the minimum max-flow among all sources to a destination. The problems of maximizing the minimum connectivity and the minimum max-flow in a hop-by-hop network are formally defined. A family of algorithms to calculate multipath routes utilizing an existing algorithm, MA Ordering, is named Maximum Alternative Routing Algorithm (MARA). Then it is proven that MARAs calculate the optimal solution for the problems. Chapter 7 proposes a new routing architecture, called the Drouting architecture, that utilizes MARAs and a packet tag forwarding architecture. The routing architecture is evaluated by simulations in terms of probability of failure recovery (i.e., the network availability). The simulations are executed on the network topologies of a few major Internet service providers inferred by the Rocketfuel project [92]. The comparison with another multipath routing called Deflection showed that Drouting architecture improves the failure recovery property. Chapter 8 discusses methods to perform traffic engineering on the Drouting architecture. A network optimization method is studied for a given traffic demand matrix. Chapter 9 concludes this research. Chapter 10 discusses the future works and prospects of the Drouting architecture and the availability of the Internet. Failure detection technologies, the difference in network operation and network debugging from the current Internet, and issues in migration and transition to the Drouting architecture are described.

Chapter 2

Highly Available Internet

This chapter defines the "highly available Internet" as the Internet with minimal functions to forecast and control the network availability. An ideal, reliable communication infrastructure to which Internet should evolve is referred to as the "Dependable Internet". The set of functions required in the highly available Internet is a most important subset required in the Dependable Internet. Hence, providing the functions required in the highly available Internet is the significant advancement in approaching to the Dependable Internet. Then, a strategy to implement the highly available Internet is explained. This chapter describes the position and perspective of this work.

2.1 Terminologies and Definitions

Numerous research projects on reliability in the distributed systems have been done in the past such as for operating system, Inter-Process Communication (IPC), programming, software, and database, though little has been done for the communication network [90, 46].

Stankovic [95] described the definitions on failures, errors, faults, and reliability:

Failure is an event at which a system violates its specifications.

Error is an item of information which, when processed by the normal algorithms of the system, will produce a failure.

Fault is a mechanical or algorithmic defect which may generate an error.

Reliability is the degree of tolerance against errors and faults.

Avižienis et. al. [8] summarized the concepts of dependability, availability, and reliability (Note that reliability is redefined):

Dependability is the ability to deliver service that can justifiably be trusted, is an integrative concept that encompasses availability, reliability, safety, confidentiality, integrity, and maintainability, and is interpreted in a relative probabilistic sense.

Availability is the readiness for correct service.

Reliability is the continuity of correct service.

Safety is the absence of catastrophic consequences on the user(s) and the environment.

Confidentiality is the absence of unauthorized disclosure of information.

Integrity is the absence of improper system state alterations.

Maintainability is the ability to undergo repairs and modifications.

In this research, the terms “failure” and “fault” are used interchangeably. In a layered communication protocol, a fault in a lower layer can be deemed as a service failure from an upper layer. Based on the above definitions, followings are defined in this research.

Network Failure is one or more losses of reachability caused by an error or a fault in the network (assuming reachability is the specification of communication network).

Failure Recovery is recovery from network failures.

Network Availability is defined as the probability for a communication to be successfully performed in the network (successful communication is the correct service).

Network Reliability is defined as the network availability over time.

2.2 Illustration of Network Availability

The notion of network availability is reviewed in this section.

Figure 2.1 illustrates the notion of network availability in the TCP/IP internet layering model [19]. As a communication session reaches from a source host to a destination via routers, the communication traverses several abstract network elements, such as the application itself at the application layer, TCP [78] instance at the transport layer, routing protocol instances and IP forwarding process at the Internet layer, Ethernet protocol at the network interface layer, and various hardware, such as network interface cards (NICs), copper cables, switching devices, wavelength division multiplexing (WDM) devices, and optical fibers, at the hardware layer. Each abstract network element has its own failure probability. For example, the failure in network interface card, the routing protocol’s failure, and also the network operator’s mistake can be considered as the failure of related network elements. The success probability for a communication, i.e., the network availability, can be expressed

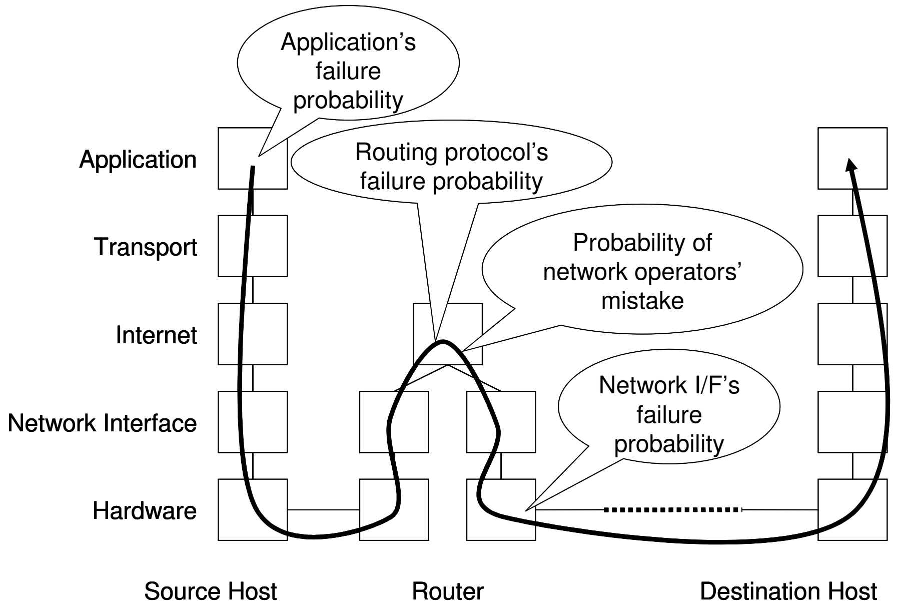
Application's failure probability
Routing protocol's failure probability
Probability of network operators' mistake
Network I/F's failure probability
Application
Transport
Internet
Network Interface
Hardware
Source Host
Router
Destination Host

Figure 2.1: Notion of network availability and failure probabilities.

as a product of all the network elements' success probability, when considering only a single path through the network.

$$P_{success} = \prod_{i \in \text{all elements in the path}} p_i. \tag{2.1}$$

2.3 Definition of the Highly Available Internet

As the first step of the research of reliability of the communication network, this dissertation focuses on the network availability, i.e., it does not consider the possibility that the success probability changes over time, and considers the probability at a fixed time. Network availability is always required, and the most fundamental, hence important, attribute in the dependability. Increasing the network availability significantly contributes to the dependability of the Internet.

As an indicative objective of this research, highly available Internet and Dependable Internet are defined below.

highly available Internet is the Internet with functions to forecast and control the network availability.

Dependable Internet is the Internet with sufficient dependability to serve mission critical services such as emergency phone calls.

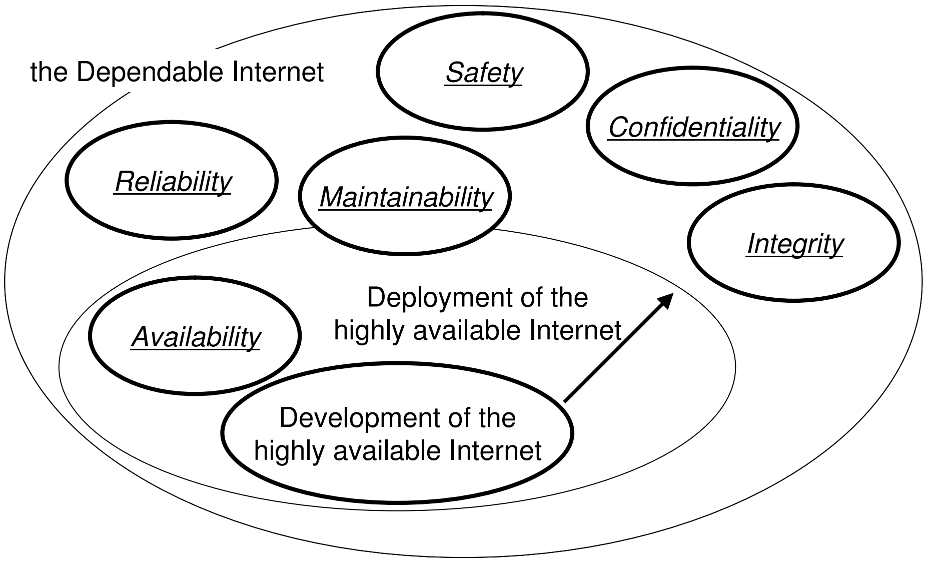

Figure 2.2: Relations of dependability, Dependable Internet, and highly available Internet.

Figure 2.2 shows the relationship between attributes of dependability, the Dependable Internet, and the highly available Internet. The "Development of the highly available Internet" is tackled in this research. When the highly available Internet is deployed ("Deployment of the highly available Internet" in the figure), availability and a part of the maintainability is supported, where maintenance of and traffic engineering in the network can be performed preserving the high availability. Reliability is not supported, in the strict definition in Section 2.1, since the highly available Internet does not take the time into account. The Dependable Internet is achieved when all attributes of dependability are satisfied for mission critical services.

2.4 Problem Statement

The main point of the definition of the highly available Internet is that network administrators should be able to increase or decrease the availability of the network voluntarily by adding or limiting the amount of investment in network resources, such as by adding a number of alternative links or by increasing the available bandwidth. Technical issues for it is that whether all the functions and technologies are provided to enable the control of availability, or not.

In the highly available Internet, one can improve the probability of success of connection establishment between a particular source and a destination, by provisioning many alternative paths between them, for example. Further improvement of availability may be afforded by replacing routers and/or circuits with higher-priced, more reliable ones. Similarly, one can save the economical cost by decreasing the availability to lower, however still forecastable, value.

The current Internet is not the highly available Internet because there are many types of problems that can be neither avoided nor automatically repaired, no matter how one invests in the network resources in advance. Such kinds of problems include:

- Hardware problems. All hardware breaks down as time elapses. Hardware failures may result in Byzantine failures [73], where detection of malfunctions cannot be expected.
- Software bugs in routers. It is very difficult to fix all software deficiencies in every possible situations encountered in real-world operations.
- Congestion in network links or on routers. Traffic is dynamic and can change drastically for various reasons, such as topology changes caused by failures of the network links, and even policy changes of neighboring networks. It is impossible to completely avoid congestion.
- Mis-configurations and mistakes in network operation by the network administrators.

For example, in the current Internet, even when one provisions many alternative paths, a router that is problematic for one of the above reasons may destroy the communication reachability despite the existence of healthy alternative paths. Consequently, in the current Internet, the availability of the network may not increase even after the provisioning of alternative paths.

However, with sufficient additional routing functionality, the current Internet can be improved to become the highly available Internet. The following sections describe the approach to improve the availability of the Internet and to achieve construction of the highly available Internet.

2.5 Objective of This Dissertation

The objective of this dissertation is to develop and provide the minimal functions that relate to routing technologies and that are required for the highly available Internet. More precisely, it is to provide the minimal functions for routing technologies to enable control of the availability of the network.

The required but not yet provided functions are discussed in Section 2.7, and are summarized in Section 2.9.

2.6 Basic Principle of the Approach

Before getting into the discussions of overall perspective on communication systems, the basic principle of our approach to improving the network availability of a communication network is described here. Failure recovery is the most important feature to improve the network availability, since if failure recovery fails, the network outage continues and the availability degrades. This research as a whole tries to make the failure recovery perform well, along with the related other features, such as network planning and traffic engineering. This section gives the principle of this work regarding how to improve network availability actually and practically. The approach described in this section is achieved through the construction of the Drouting architecture (Chapter 7).

Using a general multipath approach, network availability can be increased indefinitely close to 1.0, which means that really highly available network can be implemented. The approach in this dissertation to achieving this is two fold.

First, with sufficient multipaths, the probability that the network is totally divided and disconnected can be made indefinitely lower, as is calculated in the theory of reliability. The more multipaths, the more the probability that the network has at least one available path (i.e., connected). The first part of the approach is to increase the number of multipaths to satisfy the required network availability. Some examples of system reliability calculations with multiple paths are given in Section 2.6.1.

Second, assuming that retries of communication attempts are assigned to different paths, performing retries improves the success probability of the communication, despite the existence of failures in the network. Hence when a communication attempt is made, which network path is assigned, and at which frequency, are considered in this research. This multipath retry approach makes it possible to satisfy the required network availability, by determining a number of retries. In particular, the retry approach achieves the construction of the network with 99.999% (five-nines) communication success probability, given sufficient alternative paths and sufficient number of retries. Section 2.6.2 describes a simple example of multipath retry model to improve the communication success probability.

2.6.1 Multipath Provisioning

In Figure 2.3, nodes such as s and t represent communication devices, and links represent communication paths. There are three paths from s to t that their reliabilities are 0.9, 0.6, and 0.8, respectively. Overall system reliability from s to t is calculated as the probability that at least one of the path is functioning. Hence, the s-t reliability in Figure 2.3 is:

$$1 - (1 - 0.9)(1 - 0.6)(1 - 0.8) = 0.992 \tag{2.2}$$

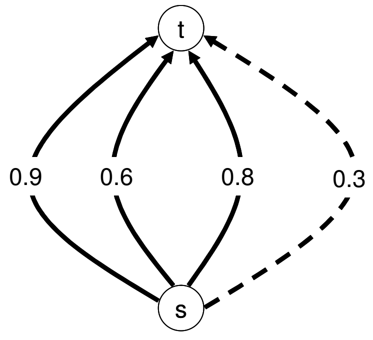

Figure 2.3: Multipath provisioning.

Adding another path, even if it has low reliability such as 0.3 (right-most dashed line in Figure 2.3), can slightly contribute to overall system reliability. After adding the path with the reliability 0.3, overall *s*-*t* reliability becomes:

$$1-(1-0.9)(1-0.6)(1-0.8)(1-0.3)=0.9944 \tag{2.3}$$

The example shown here assumes disjoint multipaths. If the multipaths are overlapping at some nodes and/or links, the complex calculation such as summarized in Appendix A is necessary.

2.6.2 Multipath Retry

In the current Internet communication model, it is not possible to utilize multipaths efficiently. TCP [78] cannot handle packets transmitted over multipaths efficiently, because these packets may be reordered, which in turn degrades the TCP performance. SCTP [96], a newer reliable transport protocol that is expected to be able to handle multipath sessions, is not yet deployed. Sending copies of a packet over the multipaths simultaneously for redundancy purpose is sometimes being considered and is called "$1+1$" in failure recovery research [55], but network operators are reluctant to do it because it consumes the bandwidth multiple times.

Because multipaths cannot be utilized efficiently, the routing and the forwarding system in the Internet pins all communications for a destination to a single network path. This fact, in turn negate the prospect in Graph theory and Reliability theory in Section 2.6.1, that "the more multipaths and more connectivity means more system reliability". Because a densely constructed graph and its multipaths cannot be used efficiently, the construction of the network graph sometimes results in a simple graph, such as a tree.

The fact that communication sessions use a single path even if multipaths are available limits the network availability, given that each path may fail with a certain probability.

To use multipaths in the Internet efficiently and practically, "multipath retry" is introduced in this research. The rule is simple: a communication session is assigned to a network path randomly, so that the retries would be assigned to different paths.

Suppose that there are problems in the network, and there are n paths from a source to a destination where m paths (out of n) are affected by the problems. Communication retries are assigned a path that has a problem with a probability $P = m/n$. For example, when 1 path out of 6 paths has a problem, i.e., $m = 1, n = 6$, every retries fails at a rate of $P = 1/6$. If an end node retries x times, the probability that all the retries fail is:

$$(1/6)^1, (1/6)^2, (1/6)^3, (1/6)^4, ..., P^x \tag{2.4}$$

This is the same discussion of the probability that rolling a dice x times results in all 1s. Note that the probability of all four retries fail, $(1/6)^4$, is 0.0007716, that is significantly small probability. This means that retrying on the multipaths up to four times will make the communication succeed in the probability more than 99.9%.

The benefits of this approach are as follows:

1. failure probability decreases drastically to a very low value when the number of retries x grows,
2. the resulting availability is forecastable to a probability, and
3. we can control the resulting availability by changing x and n (or m/n), if m and n are known.

The drawbacks of this approach are:

1. since the path is chosen randomly, the desired path may not be used,
2. the result such as the number of retries until success is not stable over time (i.e., the result is stochastic),
3. there is no apparent indication to decide whether there is no available path at all.

For the method to implement this in communication network, see Chapter 7.

2.7 Perspective

The objective of this dissertation is providing the minimal functions that are required to implement the highly available Internet. This section discusses and identifies the minimal functions which are deficient or insufficient in the current Internet.

For improving the network availability of the Internet, the issues are divided into following areas.

1. Network Planning (Section 2.7.1)
2. Availability Calculation (Section 2.7.2)
3. Maintenance of Network Elements (Section 2.7.3)
4. Problem Detection (Section 2.7.4)
5. Recovery from Problem (Section 2.7.5)
6. Traffic Engineering (Section 2.7.6)
7. Service Protection (Section 2.7.7)

For each area, a description, function required in the area to achieve the highly available Internet, and prospects using the approach outlined in this dissertation are given. This section gives the scope of this research.

2.7.1 Network Planning

Designing a communication network, including tasks such as determining the entire network graph structure, which type of router is employed, where the routers are located, and which type of datalink technologies are utilized to link them, is called "network planning". Network planning involves many aspects. It also includes capacity planning and planning of communication delay.

Most past network planning has been based on previously-given requirements. Examples for the previously-given requirements include the following:

- physical location of communication devices such as the locations of network operation centers (NOCs), network access points (NAPs), and point of presences (POPs),
- the requirements derived from limitations in datalink technologies, such as length and capacity limitations in Ethernet, synchronous optical network/synchronous digital hierarchy (SONET/SDH), and WDM, and
- economical cost to provision a circuit between POPs.

Given the requirements, past network planning tended to focus on capacity planning, and little has been considered regarding network availability [75, 48].

Since the capability to recover from failures should be primarily considered for the purpose of the network availability, the design of a network graph structure with sufficient redundancy is the most important prerequisites in this research. In order for communications to recover from failures, alternative physical communication paths are required. Constructing a network graph with sufficient redundancy is imperative. This serves as the multipath provisioning mentioned in Section 2.6.

In Graph theory, calculation of connectivity on a graph and determination of minimal additional edges required to satisfy k-edge or k-vertex connectivity have been studied [97, 62]. They can be used to construct a network graph structure that have sufficient redundancy.

Although a network graph structure with sufficient redundancy is the prerequisite for the purpose of the network availability in this research, methods to design network graphs with sufficient redundancies are beyond the scope of this research, as another separate research topic. This dissertation assumes that network graphs already contain minimal redundancy with alternative connections for most source-destination pairs, by network construction involving either graph theoretical calculation or manual check. This assumption seems plausible, at least in the networks of large ISPs, as shown in Rocketfuel [92].

2.7.2 Availability Calculation

In order to understand how availability is insufficient, we need a way to estimate and evaluate the availability of the current network. Little has been studied about the network availability for realistic, complex network model for the Internet in operation, and its routing.

System reliability is a general probabilistic measure to consult the availability of a system. Toward the highly available Internet, it is required to study the system reliability of each network in the Internet.

Current research on system reliability involves changing failure probability over time (i.e., failure probability model) for each component in the network [11]. For the first step to study the reliability of the Internet, changes of failure probability over time are not considered in this research. This dissertation assumes the failure probabilities of all components are given for the time in question, and the goal is to calculate the snapshot of the network availability for the time.

A method for forecasting the network availability taking the routing method into account is required, but does not exist yet. There has been no research activities, to the best of our knowledge, that studies forecasting of the current and future network availability, in consideration of the routing method.

2.7.3 Maintenance of Network Elements

The network consists of many components (called *Network Elements* hereafter) such as a router's chassis, routing and switching line card modules in routers, optic fibers, connectors, optical repeaters, WDM devices, the operating systems in the routers, and the software implementing the routing protocols.

Each network element has its own failure probability. For example, according to [85], some representative numbers of failures in 10^9 hours, called Failure-In-Time (FIT), are 10^4 for 10 km fiber, 10^5-10^6 for SONET equipment, and 10^3-10^4 for couplers and multiplexers. Statistics from Belcore (now Telcordia) [100] shows that FIT for optical transmitters and receivers are $10,867$ and $4,311$, respectively.

Network elements with high failure probability are problematic. For example when almost all network elements have excessively high failure probability, the network as a whole would become unavailable, no matter how the routing architecture is performing good. Individual effort to reduce the failure probability of each network element is necessary.

The tasks of making network elements robust include fixes and replacements of hardware devices and upgrades of software. They are the activities that current network operators are executing. Maintenance of each network element is the activity to keep the failure probability of each network element low. These kinds of activities are still required to maintain the highly available Internet.

2.7.4 Problem Detection

Problem detection technologies trigger routing system to recalculate the routes in the face of node failures, link failures, congestion, and deterioration of QoS performance. It is important for them to detect the problems quickly, because when they take a long time to detect, the problem lasts longer duration and the network availability is decreased.

Currently problem detection is achieved by either detection in datalink protocols or in routing protocols. Detection in datalink protocols includes Ethernet link status and SONET/SDH link alarm. Although detection functions in datalink protocols enable quick detection of link or node failures, they are limited to detecting adjacent datalink failures. Examples of detection functions in routing protocols are the Hello sub-protocol in OSPF [59] and the "timeout" timer in RIP [54]. The principle of these is simple in that it recognizes a link or node failure if messages are not heard within a duration.

There are many research directions and emerging technologies in the problem detection. For immediate detection of a failure on intermediate nodes and links, Bidirectional Forwarding Detection (BFD) [41] can be used. For detection of congestion, available bandwidth estimation techniques [45] can be used. For detection of deterioration in QoS, the detection should be done individually for each service, because separate service tends to have different QoS constraints. Setting thresholds to each communication parameters such as RTT (communication delays), bandwidth, and jitters on packet arrival might serve for the purpose. Further efficient method for problem detection is open to research.

2.7.5 Recovery from Problem

No matter how well all network elements are maintained, failures occur. When a failure occurs, communication reachability may be damaged, and communication system needs to recover from the reachability problem.

There are several technologies to recover from problems.

First, there are several failure recovery mechanisms in datalink technologies. SONET/SDH and WDM have Self-Healing Ring (SHR) restoration mechanisms (*WDM protection*) [33, 85]. Spanning Tree Protocol (STP) (IEEE 802.1d) and Link Aggregation (IEEE 802.3ad) in Ethernet technologies can also be used for this purpose, when there are redundant Ethernet links. MPLS Fast ReRoute (FRR) techniques [72] are also deemed as the failure recovery mechanisms in datalink technologies, from the viewpoint of this research. These technologies are for recovery from a failure of a single node or IP link, and thus do not achieve network-wide failure recovery.

Routing protocols, such as RIP, OSPF, IS-IS, and BGP, are those responsible to detect and recover from network-wide failures in the Internet. Failure recovery by routing protocols is also called *IP restoration*, as opposed to WDM protection described above. Although they are relatively slow, i.e., usually taking from tens of seconds to minutes, IP restoration techniques can perform network-wide failure recovery. However, there are still many failures that cannot be detected and recovered from using the current Internet routing protocols, such as software bugs and congestion, as mentioned in Section 2.4. Consequently, failures that cannot be detected by routing protocols must be repaired manually by network administrators. Hence the failures tend to be of long duration, such as an hour or even a day.

Another approach to recover from problems is an overlay network, such as RON [5] and PlanetLab [74], where distant application instances construct their own virtual networks and forward their messages based on their own routing decisions. Routing on an end application decision has benefits such as application specific route evaluation is possible, and failure recovery may be possible even when the routing protocols fail to recover, by specifying a roundabout path on the overlay network. However, this approach has several problems; 1) it complicates the communication structure and tends to result in problems in terms of extendability and efficiency, in particular, resulting in excessive amount of liveness probing packets [63], and 2) network administrators cannot predict traffic flows and making traffic engineering effective is difficult [79].

2.7.6 Traffic Engineering

Traffic engineering is an activity that manipulate the way traffic flows for the purpose of avoiding congestion in the network. Traffic flows are manipulated by use of some routing technologies, such as OSPF [30, 31] or MPLS [9, 10, 108, 16].

Traffic engineering requires knowledge of traffic demands in the network in advance. Thus, measuring or estimating traffic demands is important. The traffic matrix can be measured, for example, by using statistics of MPLS LSP [108]. Another way to estimate the traffic demand matrix from link loads has been proposed [111]. However, in general, it is very difficult to obtain traffic demand matrix because traffic is very dynamic, and because precision is required (errors in traffic matrix estimation degrade the optimality of the traffic engineering significantly [6]).

Although further research is recommended, this dissertation assumes that the minimal function required to estimate traffic demands is already provided. This research assumes that the traffic demands are given, and are precise. The consideration of errors in traffic demand estimation is future work.

Given the traffic demands, traffic engineering must be performed to avoid congestion in order to implement the highly available Internet. There are many traffic engineering research activities, described in Chapter 3.

However, the past traffic engineering research and technologies cannot be employed because they do not provide sufficient failure recovery mechanisms which are primarily required for the highly available Internet.

2.7.7 Service Protection

When considering a special service that is required to be highly reliable such as emergency phone call service, the service must be protected from other services, especially in terms of bandwidth. It is still possible that traffic of other services may interrupt the special service. This can be prevented by DiffServ [14], ALTQ [15], and bandwidth guaranteed virtual circuit technologies such as MPLS [16]. Another approach is to construct a dedicated network for the special service.

Although a failure recovery mechanism and a simple traffic engineering technique are provided in this dissertation, to realize the special service that needs to be highly available, these service protection efforts must be employed simultaneously with the technologies proposed in this dissertation.

While service protection technologies are required for the mission critical services, it is considered in this dissertation that it is possible to control the network availability without employing the service protection technologies. Hence, the service protection technologies are outside the scope of this dissertation.

2.8 Strategy

This section provides a strategy and the prospects of this dissertation to achieve the highly available Internet and to improve the network availability.

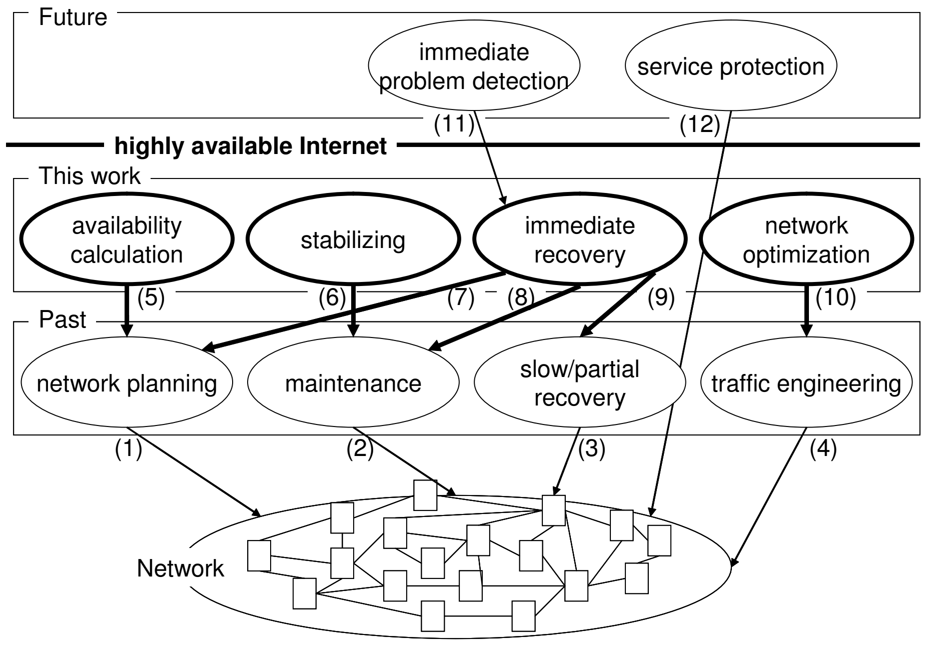

Figure 2.4: Relations of issues and strategy to the highly available Internet.

Figure 2.4 gives an overview of the past Internet (`Past`), Internet with the result of this work (`This work`), and a future Internet (`Future`) regarding the areas mentioned in Section 2.7. Relations and expected influences of each area, including expected influences of contributions of this research, are shown as the arrow lines attached with the parenthesized number. Each relationship is described in turn.

In the past Internet before this work (`Past`), Internet is constructed and operated mainly by four areas of activity: network planning, maintenance of network elements, slow or partial recovery, and traffic engineering ((1), (2), (3) and (4) in Figure 2.4). Recovery is performed slowly by routing protocols, or partially by MPLS, SONET/SDH or WDM.

The minimal functions and areas required for the highly available Internet that are deficient are, Availability Calculation (Section 2.7.2), Maintenance of Network Elements (Section 2.7.3), Recovery from Problem (Section 2.7.5), and Traffic Engineering (Section 2.7.6). They are addressed in Chapters 4, 5, 7, and 8, which achieve availability calculation, stabilizing, immediate recovery, and network optimization, respectively.

The strategy of this research is as follows.

1. Provide a method to calculate the availability of the network and to evaluate the routing system

   By providing a tool to evaluate the network availability and the routing system, network administrators and users give further consideration to practical availability of the network, and the characteristics of the applied routing system. With the tool, practical network availability that considers applied routing methods can be computed. Knowing the current network availability is the first and the requisite step to improve the availability of the Internet. The tool is expected to influence network planning in the way that the practical network availability will be considered in network planning and/or network design ((5) in Figure 2.4).

2. Demonstrate the importance of improving the availability of each network element

   Maintenance of network elements, namely activity improving availability of network elements, is not sufficient. For example, as shown in Chapter 5, OSPF routing system is vulnerable to oscillations. Other distributed systems are likely to have the similar kind of defects. By presenting the importance and a way to improve the availability of the distributed system, this work may accelerate the development of technologies which improve the availability of each network element. This work shows how to stabilize OSPF routing system in Chapter 5 and hence reinforces current maintenance of network elements ((6) in Figure 2.4).

3. Construct a new multipath routing architecture to enable failure recovery

   The basic principle described in Section 2.6 is incorporated in the new multipath routing architecture. By constructing a new multipath routing architecture, network administrators gain control on the network availability through addition of multiple paths in the network, the Internet users gain failure recovery capability, and the network becomes more robust. With the new multipath architecture, the incentive to add many multipaths in the network is enhanced, where the more multipaths the higher the network availability, hence affect network planning to provide larger number of multipaths ((7) in Figure 2.4). Since users and end hosts can recover from failures by themselves when using the Drouting architecture, the burden for network administrators of fixing the failures manually and quickly is alleviated ((8) in Figure 2.4). Finally, the multipath routing architecture enables immediate recovery in contrast to the current slow/partial recovery, reinforcing the failure recovery property of the communication network ((9) in Figure 2.4).

4. Develop a traffic engineering scheme on the new routing architecture

   By presenting a traffic engineering scheme on the new routing architecture, the prospect of achieving the failure recovery capability and traffic engineering capability at the same time is shown. The scheme enables network optimization, and anticipated to help traffic engineering activities ((10) in Figure 2.4).

5. Improve the problem detection technologies

   Application of problem detection technologies improves the failure recovery property of the network ((11) in Figure 2.4). If individual problem detection for each service is achieved, immediate problem detection for each service is possible, and the availability of each service is improved. The improvement and application of problem detection technologies are outside of the scope of this dissertation, and are given here as a future prospect.

6. Apply service protection technologies

   Finally, service protection technologies directly contribute to the communication network to protect services ((12) in Figure 2.4), thus enabling achievement of mission critical services. The application of service protection technologies are outside of the scope of this dissertation, and is given here as a future prospect.

In the step 4 in the strategy, the minimal functions for the highly available Internet are achieved. Enforcing the rest of the strategy implements the Internet that has higher availability, to the direction of Dependable Internet.

2.9 Summary

In the current Internet, practical availability of the network considering many network elements, in particular, routing systems, has not been studied. The notion of network availability is described in Section 2.2. Then the highly available Internet is defined in Section 2.1 and the goal of this dissertation is defined to provide the minimal function to implement it, in Section 2.5. Next, a basic approach, a perspective, and a strategy toward the highly available Internet are given in Section 2.6, 2.7, and 2.8, respectively.

In summary, in order to construct the highly available Internet, the first step is to be able to control the network availability. The minimal functions that are deficient in the current Internet to control the network availability are:

- A method to calculate practical network availability,
- A method to improve stability of distributed systems,
- A method to recover from any possible problems, and
- A method to execute traffic engineering in conjunction with capability to recover from problems.

These are addressed in turn in the following chapters.

Chapter 3

The Current Internet Routing

This chapter gives the description of the current internet routing as background. Descriptions include the Internet topology, the types of routing systems, the operation of the networks and the routing systems, the current trends in network statistics, and the emerging research technologies that have not yet been deployed.

3.1 The Internet Topology

The Internet is a union of networks where each network is autonomously and independently administered. Routing in the Internet is divided into two levels, above and below what is known as an *Autonomous System* (AS, sometimes referred to as routing domain); the routing internal to an AS is called *intra-domain routing*, and the routing external to an AS (in other words, the routing between ASes) is called *inter-domain routing*.

An AS was classically defined to be a network managed under a single administration using a single intra-domain routing protocol [82]. Today an AS may utilize multiple routing protocols and metrics, and an AS may even represent multiple organizations. An AS should now be regarded as just a unit of global routing with a single consistent routing policy [13]. A typical AS is administered by a single organization. An organization may manage multiple ASes, for example to improve scalability or to decrease the management cost to unite two networks. Internet Service Providers (ISPs) usually manages one or more ASes. Unless otherwise explicitly noted, the term “AS”, “ISP”, and “domain” are used interchangeably in this dissertation.

Figure 3.1 illustrates the notion of ASes. Intra-domain routing is the routing of router-level network (lower right in the figure), and inter-domain routing is the routing of AS-level network (upper left in the figure). The division of the Internet into ASes is to provide scalability of the Internet routing, because no single routing system can support flat routing (router level routing) on the entire Internet. Dividing the routing domain by AS boundary

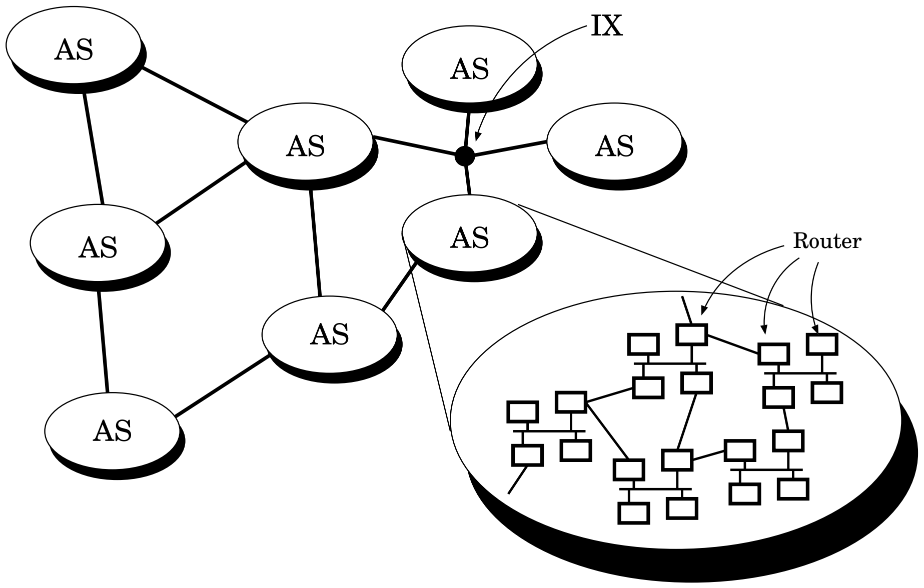

Figure 3.1: The hierarchical routing structure of autonomous systems (ASes).

also achieves locality; the routing changes inside an AS can be hidden from outside of the AS.

Around 1995, the inter-domain topology (i.e., the AS-level network graph) of the Internet was analyzed [34]. It was found that despite the growth, the diameter and degree distribution in the inter-domain topology had remained relatively constant. A new address prefix appeared every 25 minutes, a new domain (i.e., a new AS) every 12 hours, and a new inter-domain links every 8 hours. The diameter in AS units seemed relatively constant, 9 in 1994 and 10 in 1995. The growth in approximate number of ASes and IP prefixes are given in Table 3.1.

Table 3.1: Growth in number of ASes and IP prefixes.

Year/Month	approx. #ASes	approx. #IP Prefixes	Reference
1995/12	900	31,000	Govindan, et. al [34]
2008/01	27,000	250,000	CIDR Report [38]

As for the intra-domain topology (i.e., the network topology inside an AS), the Rocketfuel project [92] proposes a method to infer the network topology from outside the network. The project also publicly released the ISP network maps inferred by their method. The sizes of the biggest networks inferred by the project are shown in Table 3.2.

There have been efforts to derive appropriate models for the Internet's network topology.

Table 3.2: Number of routers and links in Splintlink, Verio, and AT&T. “BB” refers to backbone.

Name	ASN	#Routers	#Links	#BB routers	#BB Links
Sprintlink	1239	8,280	9,022	471	1,337
Verio	2914	7,284	6,490	862	1,941
AT&T	7018	9,968	10,138	487	1,067

Faloutsos et al. [29] reported the relationship between the number of nodes per out-degree and the out-degree obeyed power-law distribution in both inter-domain and intra-domain topology. Li et al. [53] argued that significant number of types of graph could exist within the power-law distribution, and proposed a intra-domain network model considering router performance.

An AS typically consists of a collection of *Points of Presence* (PoP). A PoP is a physical location where the ISP houses routers to relay connections from cities to cities. There are two classes of routers in every PoP, backbone and access routers. Backbone routers typically connect to other backbone routers in other PoPs. Sometimes backbone routers are also called core routers. Access routers connect customer networks to the backbone routers in the ISP.

As of 2004, Sprint’s PoPs connected via high-speed OC-48c (2.5 Gb/s) and OC-192c (10 Gb/s) links [39]. In Sprint PoPs, each access router was connected to at least two backbone routers, and all backbone routers were connected in a full-mesh topology to provide redundancy. For similar reasons, all Sprint PoPs were connected to at least two other PoPs. Furthermore, two adjacent PoPs had multiple parallel links connecting them, with each link terminating on a different router within the same PoP. This construction of the network ensured that there was always at least one alternative path, unless multiple failures occurred simultaneously.

3.2 Intra-domain Routing Protocols

The class of routing protocols used for intra-domain routing is called *Interior Gateway Protocol* or IGP. Typical IGPs calculate the shortest path routes depending on *routing metric*s configured on network links.

3.2.1 RIP

Routing Information Protocol (RIP) [54] is the classic and the simplest IGP. Routers running RIP collectively execute the Bellman-Ford algorithm in a distributed way (hence it is said that RIP executes *distributed computation*). The distributed execution of Bellman-Ford algorithm involves transmission of all known routing metrics (called *distance*s) to the destination between RIP routers. Because RIP routers transmit a vector of distances in each routing message, it is also called a *distance-vector routing algorithm*.

RIP only considers the distances to the destination. For a certain destination, two adjacent RIP routers inform each other of their distance to the destination. If the distances are different, the router with the longer distance will redirect its route to the destination to the router with the shorter distance. Because RIP routers compare only the distance to the destination and do not consider the path to the destination, RIP suffers from routing loops and convergence problems; sometimes RIP converges very slowly or, in a very rare case, never converges, with routing loops in the network. In order to speed up the convergence of RIP, techniques called *split horizon*, *triggered update*, and *poisonous reverse* were introduced [37]. When RIP does not converge, it increments the distance to the destination in each steps of the protocol execution. Theoretically it will never stop incrementing, and the problem is called *counting to infinity*. In order to stop the counting to infinity practically, RIP protocol specifies that the distance is limited to 15. When the distance of a route reached 16, the route is marked as unreachable and the route is deleted. Then the calculation of the route to the destination is restarted from scratch. The fact that the distance is limited up to 15 turns out to limit the application of RIP to only small networks, where the diameter is less than 16 hops. Improving the distance-vector algorithm of RIP to avoid counting to infinity has been proposed [88], but it complicates the RIP protocol specification significantly, and have never been deployed.

3.2.2 OSPF and IS-IS

Another type of routing protocols, called *Link-state routing protocol*s, are used to avoid RIP's problems in the current Internet. A link-state routing protocol distributes a complete network map among all routers, so it is classified as *distributed database* [58], compared to distance-vector's distributed computation. Open Shortest Path First (OSPF) [59] and Intermediate System to Intermediate System (IS-IS) [1] are representative link-state routing protocols. Link-state routers synchronize the map of the entire network and each router calculates routes to all destination in the network independently. The algorithm used to calculate routes from the network map is *Dijkstra's algorithm* [23], also known as *Shortest Path First* (SPF) algorithm.

The difference between OSPF and IS-IS is controversial. They are essentially the same because both are link-state routing protocols, using distributed database and Dijkstra's SPF algorithm. Three major differences are described below.

First, IS-IS was originally developed for ISO's OSI communication protocol stack, not for

IP. IS-IS can calculate the routes for both OSI and IP at the same time using the OSI protocol stack. OSPF was developed for IP and cannot calculate routes for OSI. However, sometimes it is not desired to calculate IP routes using the OSI protocol stack, because even if only an OSI protocol module fails in the network, IP routes are also affected and changed. The method to calculate routes independently, i.e., IP routes are calculated by IP routing protocol and OSI routes are calculated by OSI routing protocol, is sometimes preferred, and coined the term "*Ships in the Night*" (meaning that ships decide its direction by itself autonomously in the night).

Second, IS-IS ages routing information by decrementing the lifetime, while OSPF does by incrementing the age and limiting the upper age for routing information to survive. OSPF specifies that all routing information should be updated every 30 minutes, and to remove the routing information that is not updated for 1 hour. In contrast, IS-IS allows the originator of the routing information to decide when to update and when to remove. This flexibility influences the scalability of the routing system in that the bandwidth and the CPU time consumed in OSPF to process the distributed database grow linearly with the size of the network, while that of IS-IS can be controlled. For this reason, some major ISPs in United States of America deploy IS-IS.

Third, OSPF ensures that the routing database is synchronized with the adjacent router before advertising the link to the adjacent router as an routing information. IS-IS does not check the synchronization and just advertises the link. Hence IS-IS is said to be simpler, and OSPF is said to be complex and smarter.

All of the current routing protocols do shortest path routing. Both OSPF and IS-IS execute the SPF algorithm, and hence are shortest path routing. "Shortness" is decided by summation of all intermediate links' costs in the network path. The shortest path routing allows multiple shortest paths. In link-state routing protocols like OSPF and IS-IS, multiple shortest paths are commonly calculated, and are known as Equal Cost Multi-Paths (ECMPs). However, the number of cases where the multiple shortest paths can be calculated is relatively limited. Hence in practice the shortest path routing is basically a single path routing.

3.2.3 EIGRP

Enhanced Interior Gateway Routing Protocol (EIGRP) is another routing protocol that is actually deployed. EIGRP is a distance-vector protocol with some enhancements:

1. it implements Diffusing Update ALgorithm (DUAL) to avoid routing loops, even a transient one in routing changes,

2. it handles the composed metric: the routing metric is composed of figures of Bandwidth, Delay, Reliability, Load, and MTU, and

3. it is a proprietary routing protocol of Cisco Systems, Inc. and the protocol specification is not publicly available.

EIGRP is a single path routing protocol. Because EIGRP is proprietary, OSPF and IS-IS are more commonly used in the Internet.

3.3 Inter-domain Routing and BGP

The AS-level global Internet routing is called Inter-domain Routing. The class of routing protocols used for the inter-domain routing is called *Exterior Gateway Protocols* or EGP. Currently the only deployed routing protocol used for EGP is Border Gateway Protocol version 4 (BGP-4) [82]. A router running BGP is called a "BGP speaker".

BGP utilizes a variant of the distance-vector routing algorithm, which is called the *path-vector routing algorithm*. Path-vector routing algorithm exchanges vectors of AS-level paths, or lists of ASes, instead of distances. The path-vector routing algorithm and BGP have following advantages:

1. As route information includes a list of ASes that the routing information has propagated through, routing loops in AS-level are immediately found and rejected. This means that it is not necessary to count to infinity in order to avoid routing loops.

2. Arbitrary routing policy can be implemented by filtering routes for arbitrary reason. In contrast to a link-state routing protocol that requires synchronization of distributed databases, the distance-vector algorithm and its variants allow arbitrary filtering of routes. In a distance-vector routing algorithm family, the data traffic is routed along the path on which the routing information flows, in the opposite direction. Filtering routing information about a path only results in data traffic not being routed on the path. Utilizing this route filtering, BGP can, and indeed does, enforce the routing policy.

BGP may not be running on all routers in the domain, and BGP uses TCP to deliver BGP routing information to other BGP speakers inside the AS. This means that the intra-domain routing has to be implemented properly so that the BGP speakers can communicate each other via TCP.

To maintain the consistency regarding inter-domain routing within the entire AS, all BGP speakers in the AS must synchronize the routing information. Hence BGP handles two types of session. One is called *internal BGP* (iBGP) and is established inside the AS. The other is called *external BGP* (eBGP) and is established between ASes. iBGP is used to synchronize the routing view with the other BGP speakers in the same AS. eBGP is used to exchange routing information with other ASes.

Griffin et al. [35] discovered that conflicts on routing policies can keep entire inter-domain routing from convergence. The conflicts on routing policies are identified as “dispute wheel”, and they showed that whether inter-domain routing converges depends on the existence of the dispute wheel.

Sobrinho [91] showed that the path-vector routing requires “monotonicity” in its routing metric function to converge to a local optimal path.

BGP specifies that each BGP speaker must decide a best path among the path candidates before advertising the route to other BGP speakers. Hence BGP is a single path routing.

3.4 Network Failures

Labovitz et al. [50, 49] studied network failures in an ISP using the monitoring logs of a network management station and the routing protocol events, from November 1997 to November 1998. The study showed that approximately 20% of the router interface failures last longer than two hours.

Iannaccone et al. studied network failures in Sprint by investigating the IS-IS routing protocol event from December 2001 to April 2002 [40], and from April to August 2002 [39]. Latter reported that most failures were short-lived (called transient failures, meaning that the duration was less than 10 minutes) but 10% of the failures lasted longer than 45 minutes (called longer failures). About 70% of the transient failures were isolated failures, but less than 30% of longer failures were isolated, meaning that longer failures typically involves multiple simultaneous failures. Markopoulou et al. [56] did similar investigation with characterization of network failures from April to October 2002 in Sprint, and have found similar results.

Nippon Telegraph and Telephone East (NTT-East) experienced a major outage that lasted about 6 hours in 15th-16th May 2007 [26, 66]. During the outage, services on a part of their Asymmetric Digital Subscriber Line (ADSL) and Fiber To The Home (FTTH), as well as IP phone service including emergency phone calls, could not be established.

These indicate following facts.

1. Failures do occur.

2. Some failures last for a long time such as hours.

3. Sometimes multiple failures occur simultaneously.

4. The failures can disrupt the communication environment badly.

3.5 Failure Recovery Methods

There are two failure recovery techniques; *Protection* and *Restoration*.

Protection proactively prepares against a certain failure by provisioning a backup communication path using circuit technologies such as MPLS. Because MPLS's Label Switched Paths (LSPs) can provide Quality of Service (QoS) guarantee such as available bandwidth and estimated delays by pinning the LSP to a certain path, protection is believed to be able to provide reliable and guaranteed service. When primary path fails, the communication path is switched over to the backup path. There are a number of types in protection depending on how to prepare the backup path. The preparing of a backup path that spans only a part of primary path is called *local repair*. Allowing a backup path to reserve the bandwidth is not efficient, because bandwidth of all backup paths settled in the network remain unused. Hence sharing bandwidth between backup paths is considered, and called "$1 : N$" method compared to original "$1 : 1$" method [55].

Protection only defends against a set of predicted failures. If multiple failures occur and both primary and backup path fails, the protection technique does not provide recovery even when there are other viable alternative paths. Note that simultaneous multiple failures do occur (See Section 3.4).

Restoration is a classical method of rerouting via recalculating the routes by routing protocols. Since it is a reactive technique where the path is calculated after the occurrence of failure, rerouting by routing protocols are typically slow, in the order from a few tens of seconds to a few minutes.

Protection usually utilize a hardware mechanism (SONET/SDH alarm) to detect failures, so it can recover in a few tens of milliseconds, which is very quick. However, because protection requires preparation of backup paths for each primary path, it takes more administrative cost compared to restoration [85]. While restoration recovers slowly such as a few tens of seconds, it takes less administrative cost because it provides an alternative path using existing facilities.

3.6 Network Administration

Some aspects of Sprint's network design have been made public [39]. Due to the difficulty of installing protection paths with sufficient diversity, Sprint opted for IP-level Restoration technique using IP routing protocols, rather than protection scheme. It indicates that the following two things;

1. The protection scheme is sometimes difficult to deploy, especially in a larger network such as Sprint.

2. Improving the routing protocol is important because the networks heavily depend on it.

ISPs typically have two requirements on network administration.

1. Enough space capacity (link bandwidth) should be available, even in the event of failures.

2. Service Level Agreements (SLAs) must be satisfied. SLAs define performance guarantees that the ISPs offer their customers. Typical SLAs are upper bound of allowed packet loss rate and communication delays.

In order to achieve these requirements, the network administrators in Sprint maintain the 5 min average utilization of any link under 50%. When it grows over 50%, the link is upgraded to have sufficient bandwidth. By keeping the link utilization below 50%, congestions are avoided and SLAs are easily achieved.

3.7 Traffic Engineering

Congestion induces excessive amounts of communication delay and packet loss, and hence must be avoided. Manipulating traffic to maintain link utilization in order to avoid congestion is called *Traffic Engineering* (TE).

Traffic engineering requires preliminary measurement and prediction of traffic demands, generally as a traffic matrix. However, accurate prediction of traffic matrix is difficult because traffic is hard to predict and changes dynamically. If there are errors in the traffic matrix, the network optimality decreases drastically [6].

There are two methods to execute traffic engineering. One is to optimize a routing metric [30, 31]. The other is to use circuit-based approach using MPLS [9, 10, 108, 16].

3.7.1 Routing Metric Optimization

Since optimizing the routing weight configuration for a given set of demands is NP-hard, a number of heuristic approaches are given to gain near-optimal routing configuration [30, 31, 93]. However, they are not known to perform well on arbitrary network graphs.

Although it is not classified usually as a traffic engineering technique, a traffic-aware dynamic routing algorithm that performs metric optimization while running is proposed [12]. It defines a special type of routing metric, and changes the routing metric according to the traffic state, and demonstrated a good congestion avoidance property.

3.7.2 Circuit-based Traffic Engineering

In circuit-based traffic engineering, there are some types of method to perform traffic engineering.

First is the method which uses the MPLS Label Switched Paths (LSPs) in a static way, with settings of typically huge number of full meshed LSPs [108]. MPLS LSP is useful because it can be routed arbitrary in the network manually, and because traffic measurement is easily derived from the LSP's statistics. Static method, of course, does not adapt to the traffic.

Second is the dynamic methods, which moves traffic from LSP to LSP [28, 47], typically without knowing the traffic demands in advance.

Both types utilize the LSPs set in advance, and hence the administrative burden of huge LSPs is a demerit.

Another method is called Constrained Based Routing (CBR) [21], in which a computation server calculates a network path and establishes an LSP to respond a flow request. It can support consideration of bandwidth constraints of LSPs or the network links. However, the CBR technique is not efficient in terms of optimization of the network utilization. A problem called the unsplittable Multi-Commodity Flow (MCF) problem, which is equivalent to the optimization problem in the CBR technology, is known to be NP-complete [17]. The unsplittable MCF problem is a problem of routing of traffic flows on a single arbitrary path in the network graph without splitting, in the order received, while minimizing the maximum relative load imposed on network links.

Chapter 4

SimRouting: A Tool for Availability Calculation and Routing Evaluation

4.1 Overview

Since Internet communication is used for many purposes in modern day-to-day life, the importance of the availability of the Internet is increasing. Investigation of the availability of the Internet is the first step toward the implementation of the highly available Internet. However, little has been studied for the availability of the Internet practically, and more generally, for the practical availability of communication networks. Specifically, reliability calculation tools that consider routes calculated in the network does not exist, to the best of our knowledge. We know only the availability of the past Internet by statistical methods, and we do not know the availability of the current and near-future Internet. It is imperative to investigate how to determine the availability of the Internet.

Since the Internet is too large and heterogeneous to discuss its availability as a single system, as the first step, we focus on some of the Internet's constituent networks. Hereafter the term "network" indicates an intra-domain network.

It is necessary to understand the current network availability as the first step to improve the availability of the network. Further, it is desired to predict how much the availability will increase after some changes (such as changes in routing methods and routing metrics) are made in the network. This enables improvement of the network availability effectively.

The investigation of network availability includes understanding of the failure recovery property and the performance level of the network. Since the reachability is the primary specification of the network, whether the network is able to recover from a failure or not, and the probability of recovery, indicate the network availability. The performance level of the network, such as the existence of congestion that degrade throughput and increase communication delay and packet loss rate, is another measure.

Thus, in order to discuss the availability of the network, the necessary functions are the ability to calculate system availability of the network, the ability to obtain the overview of the current network, and the ability to predict the network state after changes of the network and its parameters.

Since current network simulation tools lack the necessary functions, a new simulation tool called *SimRouting* is developed in this chapter. The specific functions required to obtain network availability and network state for each current and modified network, are 1) to construct various networks in the simulation easily, 2) to modify the network variables easily, 3) to investigate the resulting link utilization as the network state, and 4) to calculate the network availability.

In particular, an applied routing method must be considered both to calculate network availability and to obtain the network state. The routing method is really important when we discuss the availability of the network and communications performed on it, because the routing method decides the failure recovery property and the performance of the communications. However, current simulation tools do not satisfy the required ability to investigate the availability and network state related to the routing method.

Further, to improve the network availability, the routing method must be improved. To improve the Internet routing, the evaluation of routing methods is the first step. However, evaluation of routing is a challenging task since it involves many variables such as network graphs, capacity of links, routing metric configurations, and traffic demands. No tool existed for the purpose to evaluate various routing methods, which considers the variation in these variables.

SimRouting, described in this chapter, helps users to investigate the state of the network in various combinations of network graphs, link capacities, routing methods, routing metrics, and traffic demands. In SimRouting, traffic loads in network links and the availability can be easily calculated for the applied routing method. It is expected that SimRouting contributes to obtaining the overview of the network and its state, indicating the insufficient part of the network that should be improved, predicting the network state after network changes, and evaluation of routing methods. Thus, this chapter contributes to improving the availability of the Internet by providing the SimRouting tool.

Organization of this chapter is as follows. Section 4.2 revisits the desired property of a new simulation tool. Section 4.3 describes the required feature for the new simulation tool. Section 4.4 discusses the other simulation tools as the related works. Section 4.5 describes the design and implementation of the SimRouting. Section 4.6 shows the sample usage of SimRouting to calculate network availability. Section 4.8 evaluates the SimRouting. Section 4.9 gives concluding summary of this chapter.

4.2 Need for Routing Simulation

Since the routing system applied in the network determines the network path that each traffic traverses, performance and availability of communication, and the entire network state in terms of link utilization and congestion, depend on the method of the routing system. Hence, in order to investigate the network state and the availability, it is critical to consider the routing method. However, current simulation tools do not provide satisfactory attention to routing method, for example, there is no way to investigate the network availability on the routing sub-graphs. Providing an easy way to construct a new routing method in the simulation and to evaluate the routing method are other desired, but yet not provided, functions.

Calculating the system reliability as summarized in Appendix A is another required feature for the communication network. Availability investigation by system reliability calculation on the network graph indicates the degree of robustness of the network as a whole. Additionally, calculating the availability on the routing sub-graphs provides the simple and plausible practical availability of network communication. Hence, calculating the system reliability is a required feature for our simulator.

Ease of handling of many network variables is another desired feature in simulations. Evaluating routing methods is required to employ better routing method in the Internet, and thus to improve the availability of the Internet. However, evaluating routing methods is a difficult task, because many network variables are related. These many network variables include the network graph structure, the setting of the routing metric, properties of the applied routing algorithm, capacities of each network link, and the demands of network traffic. It is not trivial to determine whether the routing method is good or bad even when the routing method exhibits good performance on a particular network configuration. Certain routing systems may work well on particular network graph structures and network traffic models. For example, suppose that a routing system R_A runs better than a routing system R_B on a network graph structure G_1. It is not uncommon that on the other network graph structure G_2, the relation between the routing system R_A and R_B changes, i.e., R_B runs better than R_A. To evaluate routing systems with considerations to their preferences on network topologies and traffic models, many combination of topology and traffic model must be tested and compared. Generally, simulations on many combinations of topology and traffic model involve an immense amount of time and effort, hence are difficult. A tool that minimizes that burden is valuable.

In particular, the ability to import network graph structures from other tools such as BRITE [57] and Rocketfuel [92] is desired. Importing the real network graph structure from OSPF's LSDB via SNMP helps to calculate the availability of a network that closely reflects the real world.

Finally, defining traffic demands, and calculations of the traffic loads on the network links are necessary to investigate the network state. Providing a sufficiently redundant network graph and a good routing method may not be sufficient, depending on the traffic demands. The traffic loads on the network links must be calculated to check whether there is a congested link that degrades the network performance significantly.

These desired features for the simulation tool are summarized as requirements in the next section.

4.3 Requirements for the Simulation Tool

The simulation tool necessary for investigating and improving the network availability requires the following features.

1. Ability to import a network graph structure from various tools.

   To evaluate a routing algorithm, it is necessary to investigate the characteristics and tendency of the algorithm on various network graphs. To study the applicability on the real network topology, it is necessary to investigate the network status on the major ISP's network topologies inferred by Rocketfuel, and on the real network topology retrieved by SNMP access to the router's OSPF LSDB. This ability to construct a graph automatically or to import from other tools is referred to as *Graph Definition* hereafter.

2. Ability to define and maintain network variables independently.

   To compare routing systems, it is necessary to change only one network variable and to compare the network status. For example, changing only the traffic demands while leaving other network variables unchanged is necessary to evaluate routing systems in various traffic demands. For another example, routing systems must be compared with different routing metrics, without changing other network variables such as network topology and traffic demands, to compare the best routing metric settings for each routing method. To achieve this, the network variables such as the network graph structure, the routing metric settings, the routes calculated by the routing algorithms, and the traffic demand matrix must be defined, maintained, and modified independently and separately. The combinations of these network variables produce various network states. In the various network states, how the routing systems behave needs to be observed. This ability to maintain network variables independently is referred to as *Independent Vars* hereafter.

3. Ability to construct a new routing method easily

In researching routing methods, a new routing algorithm needs to be constructed and evaluated in a simulation tool. In order to provide a way to construct it easily, a well-known programming language is preferred rather than the newly defined, obscure, languages. Also, the graph structures need to be accessed easily, and inclusion of the new routing method into the simulation tool is desired to be simple. This ability to construct a new routing method easily is referred to as *Develop New Routing* hereafter.

4. Ability to calculate system reliability for the network graphs

   Calculating the system reliability for a network graph is desired to evaluate the network graph. This also helps to determine whether adding a certain network link is good or not, thus contributing to the network planning activity. Also, calculating the system reliability on the routing sub-graph indicates simple practical network availability, as described in the next item. This ability to calculate system reliability is referred to as *Availability Calc* hereafter.

5. Ability to calculate routes and the routing sub-graph

   Calculating the routes using a routing method is necessary to evaluate the routing method later in terms of network loads and existence of congestion. Handling of routing sub-graphs as a network graph enables the calculation of practical availability, and the comparison of the resulting availability of applied routing method with the availability on the base network graph. This comparison helps the evaluation of the routing method. This ability to calculate and investigate on routing sub-graphs is referred to as *Routing Sub-graph* hereafter.

6. Ability to define traffic demands and to calculate network loads

   Various traffic demands needs to be defined and modified easily to investigate the preference and compatibility with the network graph and the routing methods. After loading the traffic demands along the route in the simulated network, the resulting network state must be checked for each link's utilization and the existence of congestion by calculation of network loads. This ability to define traffic demands and to calculate network loads is referred to as *Traffic Demands & Loads* hereafter.

4.4 Existing Simulation Tools

- Network Simulator (NS) [65] is a famous network simulator targeted at network research. Various TCP algorithms, queues, moves of nodes, and radio propagation models are supported. NS is written in C++, and employs Tcl scripts to describe the simulation scenario. Graph Definition in NS is relatively manual in that users should write

Tcl script, although users can create an arbitrary graph construction algorithm in Tcl. Some graph generation tools support NS, for example BRITE supports an output format to be included in NS. Independent Vars is supported in the sense that users can write separate Tcl scripts for each network variables. Develop New Routing in NS is hard because users must understand some abstract notions in NS, such as Agent and rtProto, and because Tcl interface must be provided in C++ program codes to be included in the simulation. Availability Calc and Routing Sub-graph are not supported. Traffic Demands & Loads is supported by placing traffic agents on nodes and observing the queues in the nodes.

- GloMoSim [110] is a discrete-event simulation tool targeted mainly at wireless networks. It supports mobility models, radio models, and many Mobile Adhoc NETwork (MANET) routing protocols [42], in a layered fashion. It is based on a C-based discrete-event simulation language, named Parallel Simulation Environment for Complex Systems (PARSEC) [102]. Since GloMoSim targets wireless networks, GloMoSim handles the topology as spaces, rather than graphs. Although Graph Definition can be automatically generated by inherent algorithms, importing from other tools is not supported. Independent Vars is supported in the sense that users can write separate configuration files for each network variable. Develop New Routing in GloMoSim is relatively easy since users must only write a single C module and hook `send`/`recv` functions on the global GloMoSim. However, due to the layered structure and space division approach of GloMoSim, access to the global topology seems not to be possible. Thus, routing algorithms that are not defined as distributed protocols are not easily developed in GloMoSim. Hence, Develop New Routing in GloMoSim is attributed to be hard. Availability Calc and Routing Sub-graph are not supported. Traffic Demands & Loads is supported in the sense that users must place many application entities in the network to define traffic demands.

- QualNet [87] is a commercial derivative of GloMoSim. It is written in C++, and supports many advanced features such as wired networks, real-time simulation, multi-threading, animation, and scalability of supporting 3500 nodes. QualNet has the ability to gather the network topology information from real networks via SNMP. QualNet enhances many features to simulate or even emulate the real networks, and has the ability to cooperate with a battle simulator called OneSAF[*] Testbed Baseline (OTB) [70] and with a software for the Aerospace Industry called Satellite Tool Kit (STK) [4]. However, except for the SNMP support, evaluations on the required features remain the same with GloMoSim.

[*]One Semi Automated Force (US Army Computer Generated Forces)

- OPNET [71] was developed at MIT, and released as the first commercial network simulator in 1987. OPNET supports most network protocols such as OSPF and MANET routing protocols. OPNET can import network models from commercial network management tools, such as CiscoWorks and HP OpenView. Evaluations of OPNET on the required features are similar to that of QualNet.

- OmNet++ [69] is a discrete event simulation environment written in C++. It uses the newly defined language, called NED, to organize the simulation. Although Graph Definition in OmNet++ must be done manually either by a text file or graphically, BRITE can export a graph file that OmNet++ can read. Independent Vars is not supported in OmNet++. Develop New Routing is similar to GloMoSim, in that construction of new module is easy (write C++ module and register it by `Define_Module()` macro), but constructing an routing method without distributed protocol specification is difficult, due to lack of easy way to access global topology. Availability Calc and Routing Sub-graph are not supported, Traffic Demands & Loads is supported.

- Georgia Tech Network Simulator (GTNetS) [60] is a network simulation environment for researchers. GTNetS is a set of C++ objects, and the simulation scenario is written in the C++'s `main()` function. Graph Definition in GTNetS is basically manual, but it supports importing from BRITE. Independent Vars is not supported. Develop New Routing is easy in that users can write arbitral C++ program code with easy access to the global topology. Availability Calc and Routing Sub-graph are not supported. Traffic Demands & Loads is supported.

In summary, all the existing simulators target simulating a real network state such as available bandwidth and packet loss rate, in a timed system. Their features are mainly focused on showing TCP performance and a detailed network protocol operations. Their purposes are not for the comparison and evaluation of routing systems on various network topologies, and hence they are not good at the task. This leads to a motivation of developing a new routing simulation tool, SimRouting.

The comparison of simulation tools in supported features is summarized later in Section 4.8.2 along with the evaluation of SimRouting.

4.5 Design and Implementation of SimRouting

This section describes a routing simulation tool, SimRouting, developed in this research. The notion of network variables in SimRouting is illustrated in Figure 4.1. Each network variables, such as Network Graph Definition, Routing Metric Definition, Routing Algorithm, Routes, Traffic Definition, Network State, are handled separately, from the command line interface module at the top of the figure. Supported commands and instructions for each network variable are described in the following sections.

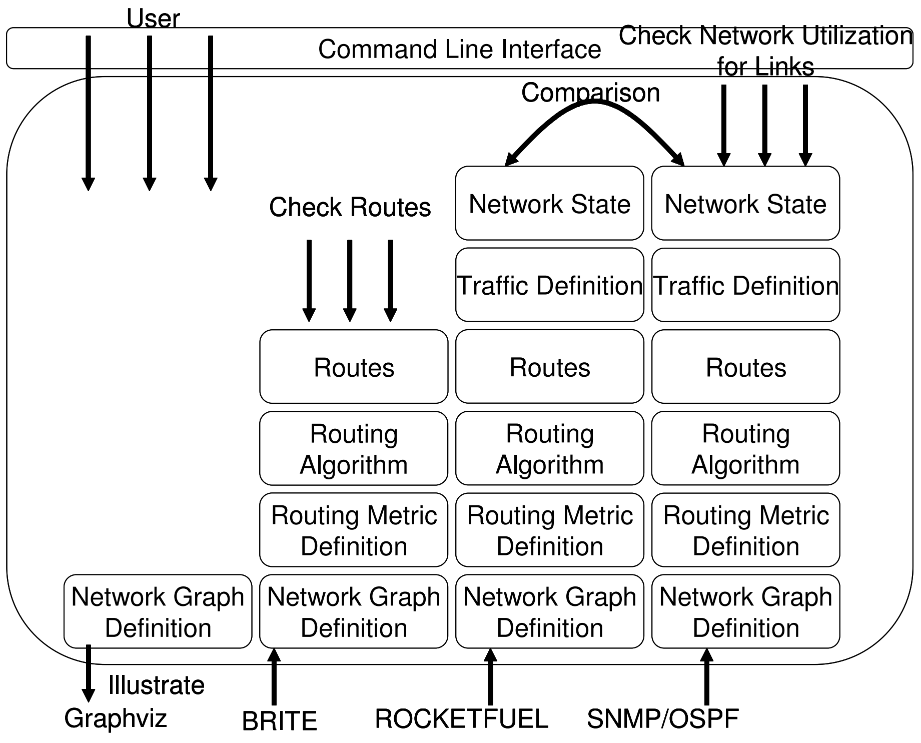

Figure 4.1: SimRouting architecture.

4.5.1 Command Line Invocation and Scenario Files

The users instruct simulation steps and network status check, through the command line interface at the top of Figure 4.1. Also, a scenario file is read as a line-oriented list of simulation commands by the command line interface module. Figure 4.2 illustrates the SimRouting interactive shell. <`cr`> and <?> are typed characters. Figure 4.3 illustrates the invocation with scenario files as arguments. Command line option `-i` specifies the run in interactive mode after execution of scenarios.

4.5.2 Network Graph Definition

Definitions of network graph can be achieved by importing from other tools such as BRITE, Rocketfuel, and SNMP [36], as well as by manual definition of each nodes and links. Figure 4.4 shows the scenario file to import graph 100, 200, 300, and 400 from BRITE, Rocketfuel maps file, Rocketfuel weights file, and OSPF's LSDB via SNMP, respectively, and

```
% ./simrouting<cr>
<cr>
simrouting> <?>
  logout           logout
  routing          enter routing node
  weight           enter weight node
  write            write information
  traffic          enter traffic node
  group            group specify command
  network          enter network node
  enable           enable features
  save             save information
  disable          disable features
  load             load configuration file
  show             display information
  quit             quit
  exit             exit
  graph            enter graph node
simrouting>
```

Figure 4.2: SimRouting interactive shell.

```
% ./simrouting <scenario-file1> <scenario-file2> ... >& <logfile>
       or
% ./simrouting -i <scenario-file1> <scenario-file2> ...
       ⋮
simrouting>
simrouting>    (can execute subsequent commands.)
```

Figure 4.3: SimRouting command-line invocation.

outputs the each of graph structure (nodes and its neighbors printed as text) to standard output.

```
graph 100
 import brite etc/topology/RTBarabasi20UNI.brite
 show graph structure
exit

graph 200
 import rocketfuel maps ../rocketfuel/rocketfuel_maps_cch/1221.cch
 show graph structure
exit

graph 300
 import rocketfuel weights ../rocketfuel/weights-dist/1221/weights.intra
 show graph structure
exit

graph 400
 import ospf localhost public 0.0.0.0
 show graph structure
exit
```

Figure 4.4: SimRouting graph importation.

SimRouting also allows generation of routing sub-graph by importing from the routes for a specified destination. It is described later in Section 4.5.4.

The defined graph can be exported as the DOT language file of GraphViz [27]. An example of importing from OSPF's LSDB and exporting to a GraphViz's DOT file with the output file name "tmp/wide-ospf-net-DATE-TIME.dot" is given in Figure 4.5.

4.5.3 Routing Metric Definition

Based on the definition of the network graph, routing metrics can be set and stored separately from the network graph definition. Note that not all routing algorithms require a definition of routing metrics. Routing Metric Definition can be set by one of following ways.

1. To set all link's routing metric as 1 (called *minimum-hop*).

```
graph 400
 import ospf localhost public 0.0.0.0
 export graphviz tmp/wide-ospf-net-%Y%m%d-%H%M%S.dot
 show graph structure
exit
```

Figure 4.5: SimRouting exportation to GraphViz.

2. To set the routing metric for a link inversely proportional to the link's bandwidth (called *inverse-capacity*).
3. To set all link's routing metric manually.

Definition of routing metrics requires definition of network graphs in advance. In the case of inverse-capacity, capacities must also be included in the graph definition. For example, BRITE generates link capacities on generating a graph. Examples of definition of routing metrics (only the part regarding routing metrics) are shown in Figure 4.6.

4.5.4 Routing Algorithm and Routes

By specifying a routing algorithm, SimRouting constructs routes on a network graph. The routes are then used to generate routing sub-graphs, or to route the traffic on the network later in Section 4.5.6. SimRouting currently supports the following routing algorithms.

1. Dijkstra's shortest path calculation [23]
2. Multipath route calculation of Deflection's (See Section 7.3 in Chapter 7)
3. MARA-MC that is proposed and described in this work (See Chapter 6)

Note that Dijkstra and Deflection requires definitions of routing metrics in advance, while MARA-MC does not.

Figure 4.7 shows examples of routing calculations. "`show routing`" command outputs the routes to standard output. All of the `routing` clauses calculate routes on the same graph, namely `graph 100`, for the comparison purpose. Deflection algorithm [109] calculates identical routes to Dijkstra, and calculates additional backup routes in a separate routing table. "`show deflection set`" command shows the contents of the separate routing table. "`routing-algorithm ma-ordering`" command specifies the use of MARA-MC routing algorithm.

Once the routes are calculated, a routing sub-graph for a specified destination can be generated as another graph. Figure 4.8 illustrates the graph generation.

```
weight 100
 weight-graph 100
 weight-setting inverse-capacity
 show weight
exit

weight 200
 weight-graph 200
 weight-setting minimum-hop
 show weight
exit

weight 300
 weight-graph 300
 weight-setting import rocketfuel ../rocketfuel/weights-dist/1221/weights
.intra
 show weight
exit

weight 400
 weight-graph 400
 import ospf localhost public 0.0.0.0
 show weight
exit
```

Figure 4.6: SimRouting routing metric specification.

```
routing 100
 routing-graph 100
 routing-weight 100
 routing-algorithm dijkstra
 show routing
exit

routing 200
 routing-graph 100
 routing-weight 100
 routing-algorithm deflection
 show routing
 show deflection set
exit

routing 300
 routing-graph 100
 routing-algorithm ma-ordering
 show routing
exit
```

Figure 4.7: SimRouting route calculations.

```
graph 500
 import routing 100 sink-tree destination 0
 show graph structure
exit
```

Figure 4.8: SimRouting importation of routing sub-graph.

4.5.5 Traffic Definition

Traffic demands are defined in a `traffic` clause in a $N \times N$ matrix (where N is the number of nodes). The only currently supported model to automatically generate traffic demands is the model of Fortz and Thorup [30]. The scaling parameter for Fortz-thorup model, α, must be specified. Another ways to set traffic are to set a traffic demand in a random and specified pair of nodes. Setting a traffic demand in a random pair of nodes is called Single-random-flow. Users can utilize Single-random-flow multiple times to produce traffic demands easily. Random seed can be specified by "`traffic-seed`" command. Since it calls `srandom()`, setting random seed may help to reproduce the same traffic demands (both Fortz-thorup mode and Single-random-flow utilize random number generations). Example traffic demand definitions are given in Figure 4.9.

```
traffic 100
 traffic-graph 100
 traffic-seed 30
 traffic-model fortz-thorup alpha 100.0
 show traffic
exit

traffic 200
 traffic-graph 100
 traffic-seed 77
 traffic-set random random bandwidth 40
 traffic-set random random bandwidth 50
 traffic-set random random bandwidth 60
 show traffic
exit
```

Figure 4.9: SimRouting definition of traffic demands.

4.5.6 Network State

In `network` clause, loading of traffic demands along routes can be instructed, and the resulting network state (e.g., link utilization of each links) can be stored. Figure 4.10 shows the example.

```
network 100
 network-graph 100
 network-routing 100
 network-traffic 100
 network-load traffic-flows
 show network
exit
```

Figure 4.10: SimRouting calculation of network state.

4.6 Example Simulation

In this section, a brief example of usage of SimRouting is shown. Dijkstra routing is calculated on a BRITE generated topology of 20 nodes. The link bandwidth capacities generated by BRITE are used in the simulation, to derive inverse-capacity metrics, and to calculate the utilization of links. The node positions generated by BRITE are also used in illustration by GraphViz, and traffic generation in fortz-thorup model. Figure 4.11 and Figure 4.12 give the output of BRITE.

Figure 4.13 shows the SimRouting scenario file for the simulation. This calculates the Dijkstra's shortest path routing with the inverse-capacity metric, on the BRITE generated topology. The purpose of the simulation is to investigate the utilization of the links after routing the traffic demands on the network. Figure 4.14 illustrates the topology. Figure 4.15 shows the traffic demands generated by the fortz-thorup model. In showing the traffic demands, the fractional parts are rounded.

Table 4.1 is the resulting network state. Note that a bidirectional link in BRITE is represented as two directional links in SimRouting. `s` and `t` denote the source and sink node of the edge (network link). `Load`, `BW`, and `Util` denote traffic loads, bandwidth capacity, and utilization of the link, respectively.

The maximum link utilization in the network was 0.930688, and the minimum link utilization was 0.004501. The average and standard deviation were 0.239890 and 0.233147, respectively.

4.7 Availability Calculation

SimRouting has the ability to calculate the system reliability of a graph. Calculation of a *s-t* reliability of a real network is given as an example of estimation of network availability in this section. The real network taken for availability calculation is the WIDE Project's

```
Topology: ( 20 Nodes, 37 Edges )
Model ( 2 ): 20 1000 100 1 2 2 10 1024^@

Nodes: (20)
0 470.00 658.00 6 6 -1 RT_NODE
1 46.00 279.00 9 9 -1 RT_NODE
2 854.00 879.00 8 8 -1 RT_NODE
3 711.00 48.00 5 5 -1 RT_NODE
4 621.00 92.00 7 7 -1 RT_NODE
5 259.00 793.00 6 6 -1 RT_NODE
6 510.00 906.00 4 4 -1 RT_NODE
7 802.00 211.00 3 3 -1 RT_NODE
8 401.00 425.00 3 3 -1 RT_NODE
9 137.00 822.00 3 3 -1 RT_NODE
10 421.00 917.00 2 2 -1 RT_NODE
11 216.00 943.00 2 2 -1 RT_NODE
12 430.00 431.00 2 2 -1 RT_NODE
13 137.00 744.00 2 2 -1 RT_NODE
14 118.00 773.00 2 2 -1 RT_NODE
15 967.00 448.00 2 2 -1 RT_NODE
16 892.00 751.00 2 2 -1 RT_NODE
17 422.00 849.00 2 2 -1 RT_NODE
18 394.00 661.00 2 2 -1 RT_NODE
19 865.00 796.00 2 2 -1 RT_NODE

Edges: (37):
0 0 1 568.70 1.90 104.16 -1 -1 E_RT U
1 0 2 443.05 1.48 888.60 -1 -1 E_RT U
2 1 2 1006.41 3.36 270.20 -1 -1 E_RT U
3 3 2 843.21 2.81 38.46 -1 -1 E_RT U
4 3 1 703.98 2.35 340.57 -1 -1 E_RT U
5 4 0 585.80 1.95 538.79 -1 -1 E_RT U
6 4 1 604.64 2.02 603.30 -1 -1 E_RT U
7 5 1 556.39 1.86 762.14 -1 -1 E_RT U
8 5 0 250.49 0.84 358.52 -1 -1 E_RT U
9 6 2 345.06 1.15 373.71 -1 -1 E_RT U
10 6 4 821.53 2.74 54.19 -1 -1 E_RT U
```

Figure 4.11: SimRouting sample BRITE generated file.

```
11 7 4 216.61 0.72 442.46 -1 -1 E_RT U
12 7 5 795.97 2.66 372.16 -1 -1 E_RT U
13 8 1 383.85 1.28 704.52 -1 -1 E_RT U
14 8 5 394.45 1.32 414.34 -1 -1 E_RT U
15 9 1 550.57 1.84 230.02 -1 -1 E_RT U
16 9 8 476.77 1.59 398.00 -1 -1 E_RT U
17 10 4 848.90 2.83 662.02 -1 -1 E_RT U
18 10 2 434.66 1.45 288.27 -1 -1 E_RT U
19 11 6 296.32 0.99 444.85 -1 -1 E_RT U
20 11 2 641.20 2.14 662.09 -1 -1 E_RT U
21 12 4 389.10 1.30 625.57 -1 -1 E_RT U
22 12 1 412.99 1.38 415.47 -1 -1 E_RT U
23 13 4 812.01 2.71 757.58 -1 -1 E_RT U
24 13 9 78.00 0.26 401.49 -1 -1 E_RT U
25 14 2 743.59 2.48 816.07 -1 -1 E_RT U
26 14 0 370.31 1.24 28.15 -1 -1 E_RT U
27 15 7 288.78 0.96 583.25 -1 -1 E_RT U
28 15 3 474.91 1.58 855.85 -1 -1 E_RT U
29 16 5 634.39 2.12 596.14 -1 -1 E_RT U
30 16 3 725.93 2.42 845.42 -1 -1 E_RT U
31 17 3 851.54 2.84 114.48 -1 -1 E_RT U
32 17 1 682.84 2.28 426.92 -1 -1 E_RT U
33 18 5 188.81 0.63 539.60 -1 -1 E_RT U
34 18 2 509.04 1.70 907.04 -1 -1 E_RT U
35 19 0 418.41 1.40 227.65 -1 -1 E_RT U
36 19 6 371.65 1.24 957.91 -1 -1 E_RT U
```

Figure 4.12: SimRouting sample BRITE generated file (continued).

```
graph 100
 import brite etc/topology/sample.brite
 export graphviz tmp/brite-dijkstra-invcap.dot
exit

weight 100
 weight-graph 100
 weight-setting inverse-capacity
exit

routing 100
 routing-graph 100
 routing-weight 100
 routing-algorithm dijkstra
exit

traffic 100
 traffic-graph 100
 traffic-seed 30
 traffic-model fortz-thorup alpha 100.0
 show traffic
exit

network 100
 network-graph 100
 network-routing 100
 network-traffic 100
 network-load traffic-flows
 show network
exit
```

Figure 4.13: SimRouting example scenario.

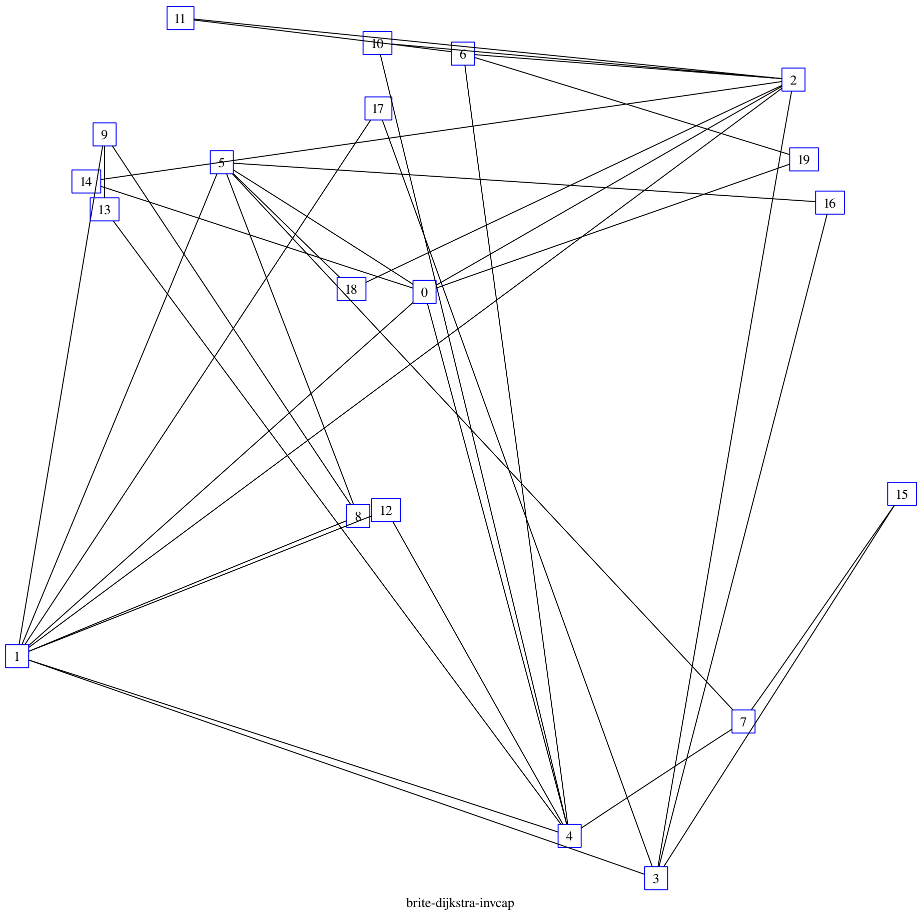

Figure 4.14: SimRouting example network graph.

Table 4.1: Results of example simulation in SimRouting.

EdgeId	s	t	Load	BW	Util	EdgeId	s	t	Load	BW	Util
0	0	1	0.000	104.160	0.000	1	1	0	0.000	104.160	0.000
2	0	2	237.665	888.600	0.267	3	2	0	274.613	888.600	0.309
4	1	2	205.539	270.200	0.761	5	2	1	179.008	270.200	0.663
6	3	2	0.000	38.460	0.000	7	2	3	0.000	38.460	0.000
8	3	1	61.894	340.570	0.182	9	1	3	19.273	340.570	0.057
10	4	0	501.446	538.790	0.931	11	0	4	232.627	538.790	0.432
12	4	1	256.082	603.300	0.424	13	1	4	316.095	603.300	0.524
14	5	1	106.838	762.140	0.140	15	1	5	195.646	762.140	0.257
16	5	0	48.893	358.520	0.136	17	0	5	1.614	358.520	0.005
18	6	2	311.449	373.710	0.833	19	2	6	277.863	373.710	0.744
20	6	4	0.000	54.190	0.000	21	4	6	0.000	54.190	0.000
22	7	4	208.373	442.460	0.471	23	4	7	145.885	442.460	0.330
24	7	5	75.087	372.160	0.202	25	5	7	10.786	372.160	0.029
26	8	1	185.276	704.520	0.263	27	1	8	279.203	704.520	0.396
28	8	5	48.881	414.340	0.118	29	5	8	58.683	414.340	0.142
30	9	1	0.000	230.020	0.000	31	1	9	0.000	230.020	0.000
32	9	8	44.699	398.000	0.112	33	8	9	50.003	398.000	0.126
34	10	4	124.623	662.020	0.188	35	4	10	309.358	662.020	0.467
36	10	2	72.375	288.270	0.251	37	2	10	116.415	288.270	0.404
38	11	6	19.351	444.850	0.044	39	6	11	64.147	444.850	0.144
40	11	2	203.272	662.090	0.307	41	2	11	379.854	662.090	0.574
42	12	4	202.114	625.570	0.323	43	4	12	51.641	625.570	0.083
44	12	1	162.272	415.470	0.391	45	1	12	19.586	415.470	0.047
46	13	4	296.558	757.580	0.391	47	4	13	283.428	757.580	0.374
48	13	9	52.602	401.490	0.131	49	9	13	117.135	401.490	0.292
50	14	2	259.766	816.070	0.318	51	2	14	180.910	816.070	0.222
52	14	0	0.000	28.150	0.000	53	0	14	0.000	28.150	0.000
54	15	7	53.833	583.250	0.092	55	7	15	44.618	583.250	0.076
56	15	3	23.640	855.850	0.028	57	3	15	28.271	855.850	0.033
58	16	5	191.373	596.140	0.321	59	5	16	14.732	596.140	0.025
60	16	3	19.491	845.420	0.023	61	3	16	92.853	845.420	0.110
62	17	3	0.000	114.480	0.000	63	3	17	0.000	114.480	0.000
64	17	1	278.302	426.920	0.652	65	1	17	133.030	426.920	0.312
66	18	5	61.409	539.600	0.114	67	5	18	455.852	539.600	0.845
68	18	2	196.119	907.040	0.216	69	2	18	205.295	907.040	0.226
70	19	0	13.709	227.650	0.060	71	0	19	114.568	227.650	0.503
72	19	6	60.210	957.910	0.063	73	6	19	239.892	957.910	0.250

Util: routing-100: max: 0.930688 min: 0.004501 med: 0.467595 avg: 0.239890 std: 0.233147

```
config-traffic-100>  show traffic
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
  0  11   7   0   1   1   1   0   3   0   1  16  11   3   7   6   0   0   5   8  11
  1   7  11   0   1  13   5   6  10  26  10  15  18   2  21   3   4   0   2   5  32
  2  10  19   0   0  10   3   3   2   3   8   1  10   6  13  11   4   0   1   7  22
  3  15  11   0   2   4   1   2   2   5   3   2  17   2   9  12   2   0  10  15  24
  4  28  17   0   2   7  16   0   8  51   4  55  30   9  26  10  11   1  11   1   7
  5  28   2   0   0   8   7   4   5  12  14  21  49   2  15  13   3   0   8  43  12
  6  30  23   1   1  11   6   4  10  35   6  56  45   4  15  29   0   1   7  37  13
  7   6  31   0   1  15  13   9   3  14  10  27   1   5  12  10   1   0  11  48  25
  8  32  19   0   1   3   4   3   2  27  13  19  35   9  16   4   4   0   1  37  27
  9  17   5   0   0   8   4   2   5  27   1   7  40   5  11   6   5   0   7   1  11
 10  18  14   0   1   4   9  12  11  20  10  44  13   7  16  10   4   1  11   3  35
 11  25   8   0   1  13   3   3   9  30   5  10  12   7  14  17   8   1   6  49  17
 12  42  19   0   2   0  23   0  20  35   2  58  10   5  24  17   1   1  14  68  27
 13  22  21   0   1   4   3   4   3  13   2  28  11   4   9  19   3   0  10  24  21
 14  31  12   0   0   2  10   3  12   5   2  28  55   0   7  16   3   0  16  44  29
 15   1   4   0   1   5   0   1   2  10   4   4  10   1   2   7   1   0   1   6  12
 16   6  12   0   0   1   4   5   5   7   4  14  12   0   8   2   5   0   4   7  14
 17  12  17   0   1  14   4  13  10  18   4  43  48   0  16   1   2   1  19  59  14
 18   4   3   0   0   0   6   1   4   5   3   6  11   3   5   1   2   0   5  10   2
 19   1   0   0   0   3   5   1   3  10   0  16  19   1   5   3   1   0   4   2  15
```

Figure 4.15: SimRouting example traffic demands.

network [107]. It is illustrated in Figure 4.16.

The availability calculation technique implemented in SimRouting is I_VT [101], one of the family of SDP with MVI. The theory is summarized in Appendix A. Although the methods of path enumeration and sorting of path list are important for the performance of the calculation (e.g., computation time), path enumeration is performed by simple depth-first-search (DFS) each time in the graph, and the sorting of path list is not performed. These means that performance improvement is possible for the computation time presented below.

The source (`s`) and the sink (`t`) of the *s*-*t* reliability are the node 5 and the node 0, respectively. The scenario file used is shown in Figure 4.17. There are 3031 distinct paths from node 5 to node 0, where the minimum number of hops is 7, the maximum number of hops is 34, and the average is 20.632794.

"`calculate reliability`" commands in Figure 4.17 instruct the availability calculations. Calculating reliabilities were executed on the base graph, the dijkstra's routing sub-graph, and the MARA-MC's routing sub-graph (described later in Chapter 6) to see how shortest path routing degrades the network availability. All the link reliability is set to 0.9 to emphasize the difference. The statistics of the calculations and the availability results are shown in Table 4.2. Note that the shortest path routing employed in the current Internet significantly decreases the availability, despite the sufficient number of network paths.

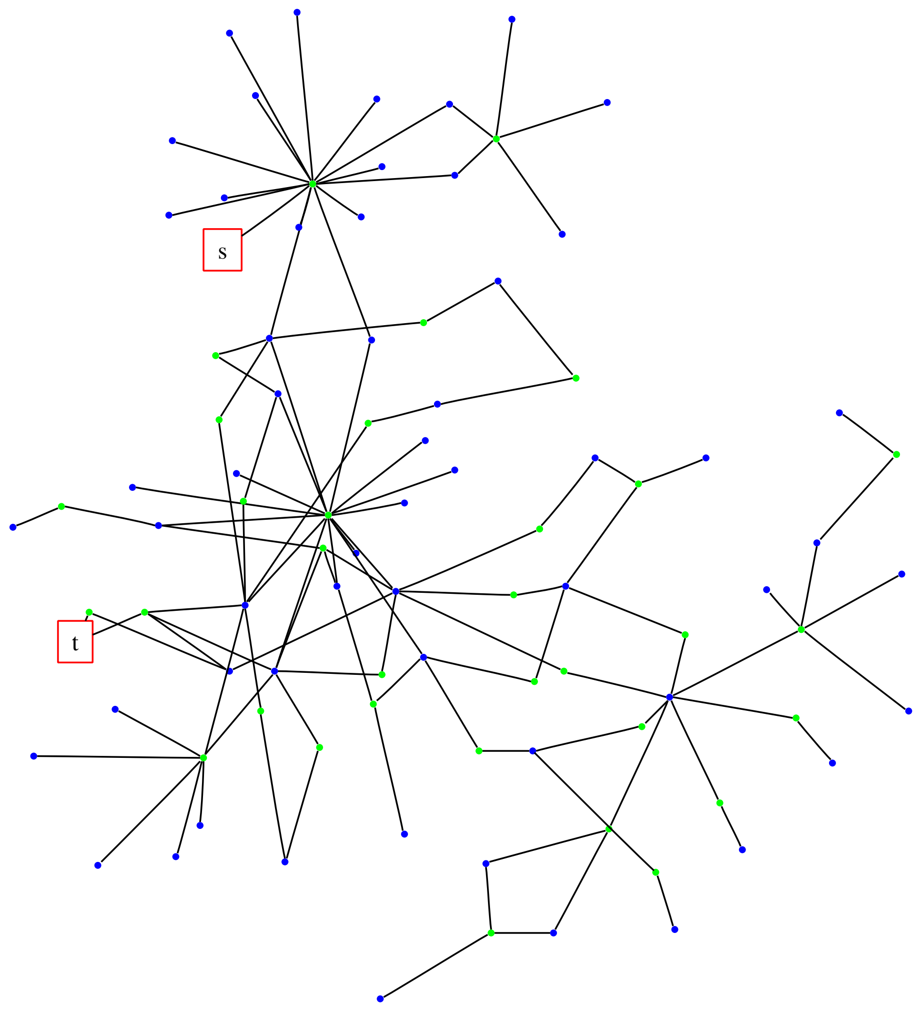

Figure 4.16: *s*-*t* reliability calculation in the WIDE Project's network.

Table 4.2: Results of the *s*-*t* reliability calculation in SimRouting.

Graph used	#Paths	#terms in I_VT	Time Taken (sec)	Availability
Base Graph	3,031	208,843	16,649	0.8641323325
Routing Sub-graph (Dijkstra)	1	1	0.000149	0.5314410000
Routing Sub-graph (MARA-MC)	6	8	0.004171	0.7963936209

```
graph 100
 import ospf localhost public 0.0.0.0
 link all reliability 0.9
 show path source 5 destination 0
 calculate reliability source 5 destination 0 stat
exit

weight 100
 weight-graph 100
 import ospf localhost public 0.0.0.0
exit

routing 200
 routing-graph 100
 routing-weight 100
 routing-algorithm dijkstra
exit

graph 200
 import routing 200 sink-tree destination 0
 link all reliability 0.9
 show path source 5 destination 0
 calculate reliability source 5 destination 0 stat
exit

routing 300
 routing-graph 100
 routing-algorithm ma-ordering
exit

graph 300
 import routing 300 sink-tree destination 0
 link all reliability 0.9
 show path source 5 destination 0
 calculate reliability source 5 destination 0 stat
exit
```

Figure 4.17: SimRouting availability calculation scenario.

Table 4.3: Supported methods in SimRouting.

Graph	Routing metric	Routing algorithm	Traffic model
BRITE	minimum-hop	Dijkstra	Fortz-thorup
Rocketfuel	inverse-capacity	Deflection	Single-random-flow
SNMP/OSPF		MARA-MC	

4.8 Evaluation

4.8.1 Number of Methods to Construct Network Graphs

Table 4.3 summarizes the options for each network variable. Only Dijkstra and Deflection can consider the setting of routing metric, hence

$$3 \times (2 \times 2 + 1) \times 2 = 30 \tag{4.1}$$

combinations of network setting can be constructed with only a few tens of lines in the scenario file. This feature can contribute a network researcher to evaluate routing algorithms and/or network settings with comparison to other similar settings. Changing only one network variable between two setting presents the influence of the changed network variable to the network state.

Note that the number of network types above (i.e. 30) is the number of method to construct a network setting. The number of actual network setting is infinite even when we limit the importation of network graph from BRITE.

4.8.2 Achievement of Required Features

Comparison of features of SimRouting with other simulation tools is given in Table 4.4. The required features are from Graph Definition (shortened as `Graph` in the table) to Traffic Demands & Loads. The rationale for the required features is explained in Section 4.3.

A few tools support importation of network graph (i.e., Graph Definition) from other tools. Note that, since OPNET can import real network status from some vendors' network management tools, its `Graph (SNMP/OSPF)` is attributed to yes (support). However, SimRouting supports the largest methods to import network graphs, which is the desired property of a simulation tool for network researchers. Support of handling of network variables independently (Independent Vars) is also supported in SimRouting, as well as other simulation tools. Develop New Routing is easy in SimRouting due to the easy access to the global network graph structure. SimRouting implements a method to calculate system reliability, while others does not. This is shown in the row of Availability Calc. The ability

Table 4.4: Comparison of features with other simulation tools.

Feature \ Name	NS	GloMoSim	QualNet	OPNET	OmNet++	GTNetS	SimRouting
Graph (BRITE)	Yes	No	No	No	Yes	Yes	Yes
Graph (Rocketfuel)	No	No	No	No	No	No	Yes
Graph (SNMP/OSPF)	No	No	Yes	Yes	No	No	Yes
Independent Vars	Yes	Yes	Yes	Yes	No	No	Yes
Develop New Routing	Hard	Hard	Hard	Hard	Hard	Easy	Easy
Availability Calc	No	No	No	No	No	No	Yes
Routing Sub-graph	No	No	No	No	No	No	Yes
Traffic Demands & Loads	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Timed System	Yes	Yes	Yes	Yes	Yes	Yes	No
Detailed Protocol Impl	Yes	Yes	Yes	Yes	Yes	Yes	No

to handle Routing Sub-graph is unique to SimRouting. All tools in the comparison support Traffic Demands & Loads.

To perform fair comparison, two measures that are deficient in SimRouting have been added in Table 4.4. These are *Timed System* and *Detailed Protocol Impl*.

Considering time in simulations is necessary to comprehend the difference, dynamics of a system over time. In contrast to the fact that most simulation tools support timed simulation scenario, SimRouting does not support simulation of time. This deficiency leads to the difficulty to handle time related variables, such as hardware failure probability (reliability of a hardware changes over time such as widely known as the bath-tub curve). In SimRouting, only a snapshot of a time can be taken into consideration. Thus, the failure probabilities of all network components at the snapshot time must be calculated manually in advance to the simulation.

Detailed Protocol Impl means whether the detailed protocol implementation is included in the simulation. Most other simulation tools take the approach of layered communication structure, where many detailed implementations of network protocols can be involved. The advantage of this approach is that the state of the network becomes closer to that of reality, because the control traffic of the underlying network protocols are also considered. In SimRouting such overhead of the underlying network protocols cannot be considered in simulation. SimRouting is intended to be as simple as possible, to provide test environment for abstract ideas in theory.

In summary, SimRouting provides the simple test environment for new abstract ideas in networks and routing. In contrast to most other simulation tools that take layered communication protocol stack approach for the purpose to simulate the real network closely, SimRouting takes the approach of abstract network such as done in Graph theory. The abstract graph approach helps to understand the overview of the network, rather than details, and therefore more appropriate in searching new technology that has pervasive influence over the entire network, such as routing algorithms. Support of all required features in SimRouting enables comprehensive study of a new routing algorithm.

4.9 Summary

In order to improve network availability, thorough investigation must be done with various network settings. Particularly, routing systems including the methodology to set routing metrics are really important in terms of both failure recovery and congestion avoidance.

However, current researches have evaluated their methods only on very limited network settings. Testing a routing algorithm on different network topologies was a difficult task because it was difficult to utilize the network model that is generated by other tools. A method that enables the evaluation of routing systems by comparison with other network setting was necessary.

A routing simulation tool named SimRouting was developed to help investigation and evaluation of various network settings. It supports the many network settings, such as by importing the network graphs from BRITE, Rocketfuel and SNMP/OSPF. The routing settings can be chosen from minimum-hop or inverse-capacity. The routing algorithm options are Dijkstra, Deflection, and MARA-MC. The traffic model supports Fortz-thorup model and Single-random-flow. The SimRouting tool supports the execution of simulation on various network settings easily, and hence contributes to improve routing systems.

This work contributes to improve the availability of the Internet by presenting a method to compare, evaluate and improve routing systems. The deficiency and tendency of routing systems can be shown by SimRouting, and hence it is utilized to improve routing systems. Improving routing systems certainly contributes to Internet availability.

This chapter discussed an evaluation of the availability of the Internet, with relation to the network settings and routing systems. Following chapters discuss how to improve the routing system actually. Next chapter discusses how to stabilize a routing system.

Chapter 5

OSPF Flap Damping

In this chapter, a method to stabilize a routing system is described, in order to improve the availability of the routing system. In Section 2.7.3, we discussed the need for improving the reliability of network elements. Here, we use OSPF flap damping as a representative example.

5.1 Overview

The communication data paths in the Internet are controlled by routing protocols. They are responsible for finding alternate paths/routes in the face of network failures, such as hardware failures, excessive load conditions, incorrect implementations, and network upgrades/routine maintenance. In the case of such failures, communication across the network can be partially or even totally lost until the routing protocols find the next best paths for all the routes traversing the failed network.

When a network event causes routes to either fail or become available, routers distribute routing update messages that permeate networks, stimulating re-computation of optimal routes and eventually causing all routers to agree on these routes. The process from network changes to recovery of communication paths is called "convergence" wherein all the routers agree on their view of optimal paths.

During convergence, there can be routing loops in the network and a subset of destinations will be reachable via sub-optimal paths or will not be accessible at all. Routing loops themselves may delay the overall convergence process as some routing control packets may be lost. Thus it is desired that the routing protocols in the routing domain take minimum time to converge, because this will lead to increased network availability since the periods for which routes will not be available nor optimal will be minimum.

The central symptom of route instability is the disappearance of a route that previously existed in the routing table. Such routes may disappear and reappear intermittently, a condition referred to as “flapping”. Thus the number of route flaps over a period of time characterizes the intensity of the perturbation in the network.

In this chapter, the effect of such persistent “route flaps” on a link state routing protocol, OSPF [59], is studied. It is observed that how the route flaps disrupt the routing and affect the communication environment adversely. Further, a scheme for damping the route flaps is implemented, and it is shown that how the methodology can solve such problems in OSPF.

5.2 Effects of Route Flaps

In this section, the effects of persistent route flaps in OSPF and a UDP [76] communication channel are explored through an experiment.

5.2.1 Network Configuration

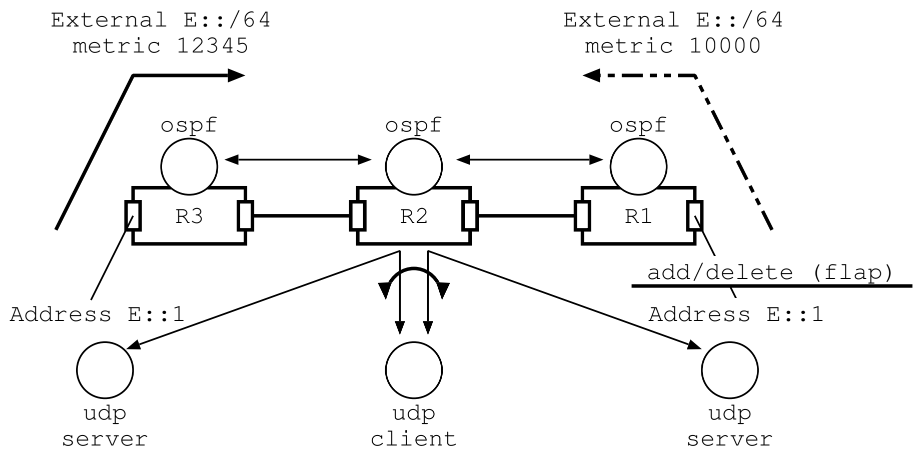

Figure 5.1: Network configuration for the route flap experiment.

The network setup used in the experiment is shown in Figure 5.1. There are three routers, each of which runs OSPFv3 [18] using Zebra ospf6d [99, 67]. Both *R1* and *R3* advertise the prefix *E::/64* as AS-External route with metrics 12345 and 10000 respectively. Additionally both *R1* and *R3* are configured with address *E::1* on their interfaces and thus each can receive packets destined to *E::1*.

R1 and *R3* are configured as *UDP server*s while *R2* runs a *UDP client* sending a stream of test UDP packets. These packets will be received by either *R1* or *R3*, depending upon whichever is chosen as the best path at that particular instant. Under the steady state conditions *R1* is the preferred destination as it is advertising the prefix with a smaller metric.

Upon receiving the test UDP packet, the *UDP server* writes down the value of HopLimit (TTL), fills its hostname as means for identification and sends it back to the source (*UDP client*). The *UDP client*, by comparing the time when a packet was sent and the time when its response is received, computes the total round trip time (RTT) for each packet. By examining the hostname filled in the packet it can know which router received that particular packet and can thus estimate when the traffic starts using the secondary path. By looking at the sequence numbers, it can determine the number of packets which get dropped. By examining the round trip hop count, it can know when the packets got stuck in a routing loop.

As described above, under the steady state conditions the primary path for all traffic destined to *E::1* is via *R1*. This is the primary path. When this path is not available, the traffic shifts to the secondary path, which is through *R3*. Ideally, the routing to the destination should stick to the secondary path when the primary path has some trouble, such as flaps in this work.

Flapping is artificially introduced in the setup by adding and deleting the address *E::1* to/from *R1*'s interface using the ifconfig command in NetBSD [98]. Zebra ospf6d is dynamically notified of this event and it in turn advertises this information to both *R2* and *R3* by originating and purging the corresponding AS-External-LSA.

5.2.2 OSPF Behavior

There are 2 built-in fixed timers in OSPF which implicitly damp the origination of LSAs against flapping entities. The first is MinLSInterval specifying that an LSA cannot be originated within 5 seconds if a previous copy was originated within this time period. The other is MinLSArrival timer that does not accept newer LSAs if a previous copy was received within 1 second back. These two timers can, however, adversely affect in certain circumstances as described below.

Figure 5.2 illustrates the relationship between the periodic *Up*/*Down* events, OSPF send/receive events for LSAs, and reachability of UDP communications. In the figure, the lowest line in the chart indicates the event causing the network topology change, which is in this case, addition and deletion of address *E::1* at *R1*. For the sake of brevity, adding the address *E::1* to *R1*'s interface is called as the *Up* event hereafter, and deleting it as the *Down* event. Each *Up*/*Down* event is generated periodically after every 1 second, as shown in the figure.

The second line from bottom shows the origination of appropriate LSA for prefix *E::/64* at *R1*. Each LSA can only be originated after every 5 seconds as the LSA origination interval is limited by MinLSInterval seconds in the OSPF specification. Thus at each *Up* event, ospf6d tries to originate the corresponding AS-External LSA with age 0 unless affected by the MinLSInterval (described as "*Active Sent*" in Figure 5.2). At each *Down* event, ospf6d tries

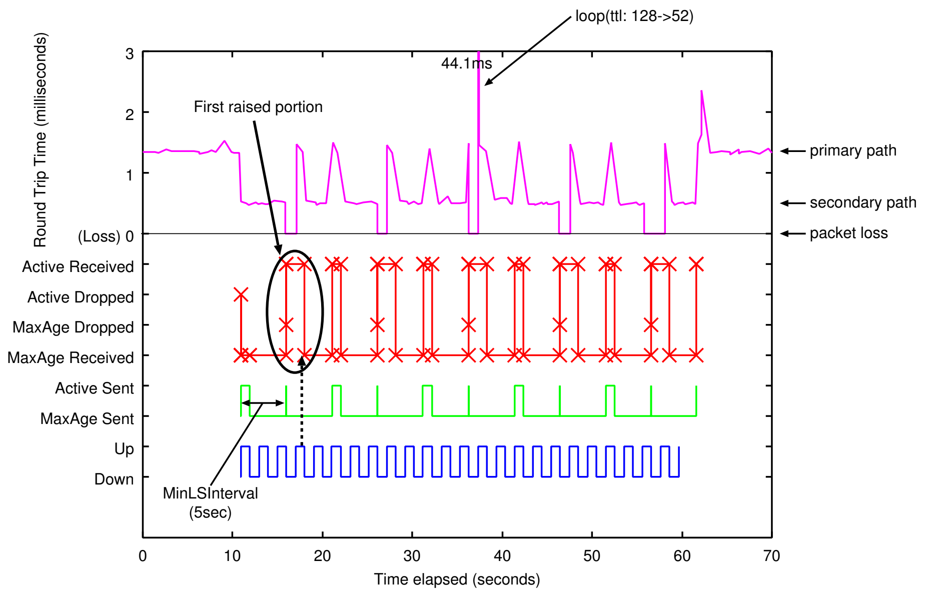

Figure 5.2: RTT, OSPF send/receive events during the flapping of 2 seconds cycle (1 second for *Up*, 1 second for *Down*).

to flush this LSA from the routing domain by flooding a copy of this LSA with its LS Age set to MaxAge (3600) (described as "*MaxAge Sent*" in the figure). It was often observed that the LSA origination was immediately followed by the attempt of flushing it. This is caused by the timing of the *Down* event.

The third from the bottom illustrates the corresponding AS-External-LSA receive events at *R2*. There are four kinds of receive events: *MaxAge Received*, *MaxAge Dropped*, *Active Dropped* and *Active Received*. As the names suggest, the events *Active Dropped* and *MaxAge Dropped* indicate the events when ospf6d rejects the received LSAs because of the MinLSArrival restraint, while *MaxAge Received* and *Active Received* indicate the events when the LSAs are accepted.

The top line in the chart indicates the total round trip time (RTT) between the *UDP client* and the *UDP server*. The packets which are lost during the re-route period are shown with RTT value of 0. The use of the primary path exhibits larger RTT, and the secondary path exhibits smaller RTT. The figure shows that route flaps affect the reachability adversely where the route periodically shift between the primary path, the secondary path, and packet loss.

In the same figure, the *first raised portion* of OSPF receive events shows the adverse effects of the MinLSInterval and MinLSArrival timers. The focused version of the figure, given in Figure 5.3, illustrates the part in detail. It is explained as follows.

When receiving the LSA that corresponds to *Active Sent* (which was delayed by *R1* due

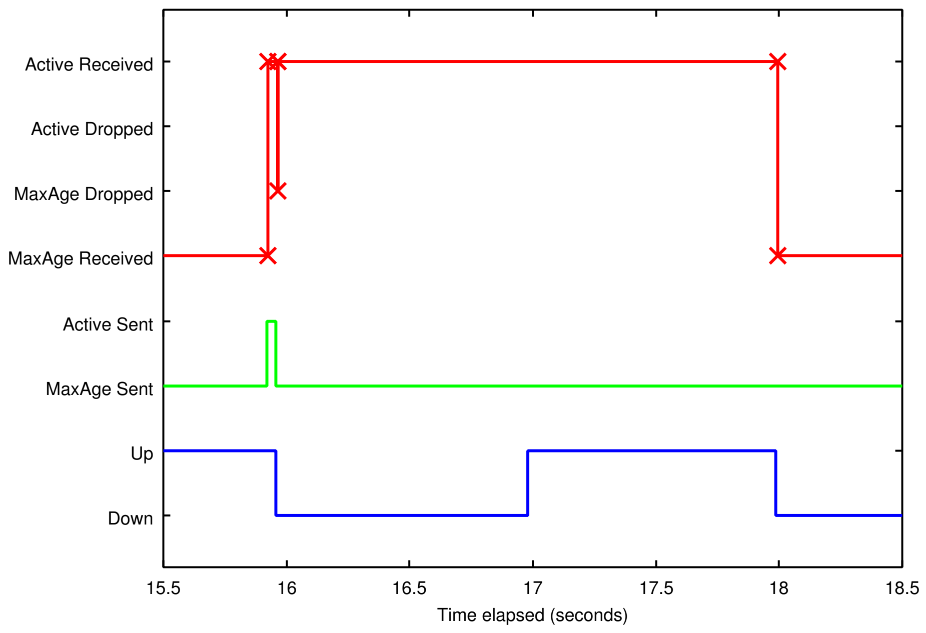

Figure 5.3: The first raised portion of Figure 5.2.

to MinLSInterval), the event *Active Received* occurs. Then, as mentioned before, the event *MaxAge Sent* gets triggered due to the real *Down* event immediately after the *Active Sent* event. Since OSPF specifies that the LSA must be dropped if a previous copy is received less than MinLSArrival (1 second) ago, this MaxAge LSA flooding is rejected by *R2*.

This results in routing disparity, where different routers have different views of the optimal paths. The packets get dropped in the test UDP communication network, because *R2* still believes that *R1* is the best path for *E::/64*, whereas in reality it is actually down.

Zebra ospf6d does not apply MinLSInterval when flooding the MaxAge LSAs, and these LSAs are flooded as soon as the *Down* event occurs. The problem is that the other routers will still apply the MinLSArrival timer. Since the other routers had just received the LSA, they ignore this MaxAge LSA (without acknowledging it). The faulty routing thus persists in the routing domain till either the next real *Down* event or the expiry of the Retransmission timer occur. In this case the event *Down* occurs first and the MaxAge LSA is originated again by the *R1*. At this time the other routers accept this LSA, the routing gets converged, and the packets start flowing through the secondary path. Even if implementations apply MinLSInterval to control the flooding of MaxAge LSA, the problem is only going to get worse, because the period during which false routing exists will last longer.

Figure 5.4 shows the worst case scenario for this environment by executing *Up* event immediately followed by the *Down* event. with a period of 7 seconds. Thus the period of *Up* event is a little (less than 1 second) and that of the *Down* event is 7 seconds.

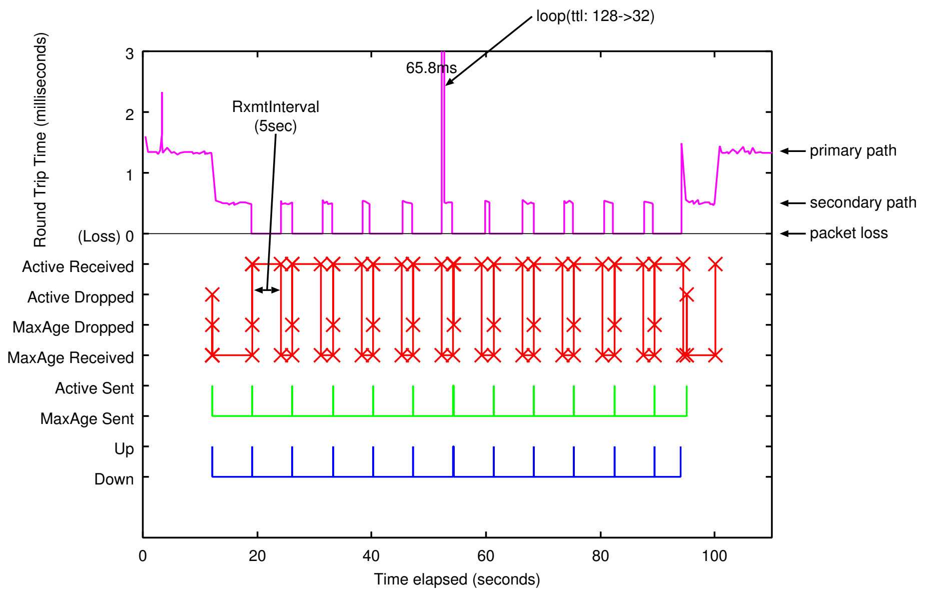

Figure 5.4: RTT, OSPF send/receive events during the flapping of 7 seconds cycle (*Up* immediately followed by *Down* and remains *Down* for 7 seconds).

This result shows that all the *MaxAge Sent* triggered by *Down* event were rejected by *R2*, so the incorrect routing information lasts until *R1* retransmits its un-acknowledged LSA to *R2* after the expiry of RxmtInterval. In OSPF, RxmtInterval a is configurable interface parameter, which is set as 5 seconds for this experiment. Here the RTT for the UDP communication channel at the top of the figure shows the packet loss for 5 seconds, which then recovers to use the secondary path until the beginning of the next *Up*/*Down* cycle, for 2 seconds (shown in the figure as the reachability via the secondary path between packet losses).

5.2.3 Experiment Result

In this section, the summary of observations in the experiment is given.

First, the current timers in OSPF (MinLSInterval, MinLSArrival), which are considered to be enough to damp short flaps, are inadequate and fail miserably when the period of flap becomes greater than 5 seconds. Even for flaps with periods less than 5 seconds there can be transient black holes depending upon how the MinLSInterval and purging of the LSA (*Down* event) are synchronized.

During the period that OSPF advertises and calculates these false routes, there are little chances for communications to use the alternative stable route even if there exists. When used in conjunction with Traffic Engineering, OSPF may further aggravate the situation because each time the network topology changes, traffic engineered paths may need to be rerouted. Even if alternate paths have been recomputed, there are overheads involved in tearing down the old path and setting up the new one.

It is shown that MinLSInterval which delays the origination of LSA up to 5 seconds and MinLSArrival which delays receiving and processing of LSA up to RxmtInterval seconds are both pathological in some particular scenarios.

By varying the period and cycles of flaps, further interesting statistics were collected. Here the total period of flap is set to 2 seconds: 1 second of *Up* state followed by 1 second of the *Down* state. Then the test UDP communication resulted in the following 10 second repeated cycle: 1 second of packet loss, 1 second of using the primary path, 3 seconds of using the secondary path, 1 second of using the primary path, and 4 seconds of using the secondary path (all these measurements are approximate). Thus in a cycle of 10 seconds the primary path (which is defined to be the best path) was only used for 2 seconds.

Given that today the IGP is used extensively, this behavior of OSPF against flaps is clearly inadequate and warrants closer inspection.

5.3 Application of Flap Damping to OSPF

To solve the problems described above, a solution very similar to one explained in BGP Route Flap Damping [103] is implemented in this section. Paths and routes can be categorized either "well-behaved" or "ill-behaved". A well-behaved route shows a high degree of stability during the extended period of time. On the other hand, an ill-behaved route experiences a high level of instability in a short period of time.

5.3.1 Flap Damping Algorithm

A figure of merit named *penalty* is maintained for each route, and it is incremented each time the route flaps. This gives the degree of instability of that route. *penalty* is increased by a constant *DefaultPenalty* when a *Down* event occurs for the route. When the route flaps, advertisements of the availability of the path are kept suppressed. When the *penalty* goes beyond a configured *Suppress* value, the route is kept from being advertised. During the period without a flap, the *penalty* continues to be decayed exponentially at a rate, where the *penalty* is reduced by half for each configured *HalfLife* seconds. When the *penalty* falls below a configured *Reuse* value, the route that has been suppressed is now advertised and the router originates an AS-External-LSA with age set to 0.

For the subsequent experiments, the parameters *DefaultPenalty*, *Suppress*, *Reuse* and *HalfLife* are set to 1000, 3000, 2000 and 20, respectively.

5.3.2 Experiment Result

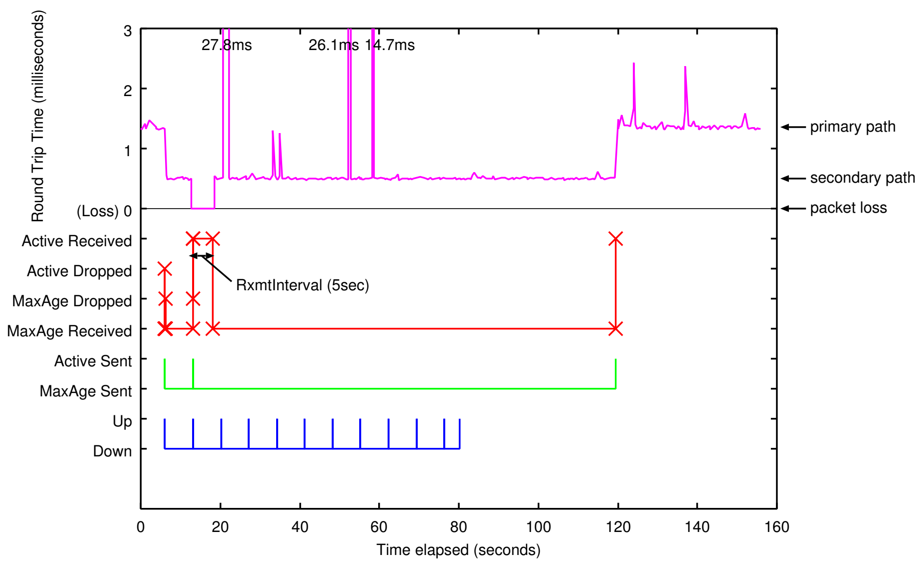

Figure 5.5: RTT, OSPF send/receive events during the flapping of 7 seconds cycle (*Up* immediately followed by *Down* and remains *Down* for 7 seconds) with flap damping.

The technique was applied over the worst case scenario when there is a flap cycle of 7 seconds in which the *Up* state lasts for a very short period of time (< 1 second), and the *Down* state lasts 7 seconds. The results are shown in Figure 5.5.

The lowest line shows how the route was flapped. It was brought *Up* for a very short period of time and was brought *Down* for the next 7 seconds. The second line from the bottom in the chart shows the OSPF send events which were significantly improved even in case of persistent real *Up*/*Down* events (i.e., when the route is flapping). RTT of UDP communication (top line in the chart) shows that while damping was performed, there were no packet losses or routing loops formed, and the UDP communication peacefully used the secondary path and was oblivious to the flapping (from time$(x) = 20$ to 80). After the flapping of the address at *R1* was stopped, it took approximately 40 seconds (from time$(x) =$ 80 to 120) of stability for *R1* to decide advertising itself as the primary path. The UDP communication was then restored to the primary path.

The *penalty* transition is shown in Figure 5.6. It shows the relationship between the *Up*/*Down* event, the *penalty* value, and the OSPF behavior of originating LSAs. At each *Down* event, *penalty* is increased by 1000 (referred to as *DefaultPenalty*) while decaying is applied simultaneously. Once the *penalty* reaches a value of 3000 (*Suppress*) the damping starts. The damping function maintains the route's state as "Down" during the period of damping, and *R1* stops any further origination and flushing of the corresponding AS-

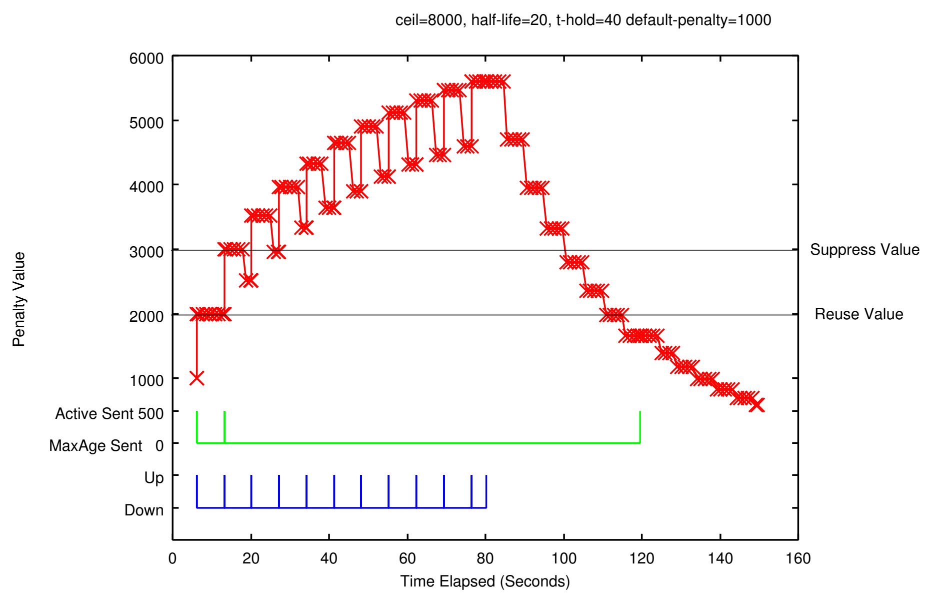

Figure 5.6: *penalty* transition during the flapping of 7 seconds cycle (*Up* immediately followed by *Down* and remains *Down* for 7 seconds) with flap damping.

External LSA for this prefix. If the flapping stops, *penalty* is not increased any more and it only decays as the time elapses. When the *penalty* falls below 2000 (*Reuse*), it is assumed that the route is now stable and an AS-External-LSA is originated for this prefix.

The reason why the LSA was not originated immediately when the *penalty* falls below *Reuse* limit, is because of the time granularity used in the implementation. The time granularity is a constant *DeltaReuse* which is set to 10 seconds.

It is possible to configure flap damping to handle either short term severe route flaps or even milder (long term) repeated route flaps, by changing the parameters. A less aggressive suppression can be applied to the case where no alternate path exists. In the simplest case, a more aggressive suppression should be applied if any alternate route/path exists.

5.4 Summary

The behavior of OSPF against flaps has been investigated in this chapter. It is shown that how flaps adversely affect OSPF routing and the communication environment in general. The current OSPF fixed timer damping function is not “flap tolerant” since each flap may lead to a few seconds of lost connectivity. MinLSInterval and MinLSArrival timers can sometimes adversely affect the network convergence. Further, because OSPF advertises and computes SPF based upon incorrect information, there are less chances of switching to the alternative stable route even if one exists.

A route flap damping technique is implemented to originate AS-External-LSA only when it is expected that the route redistributed into OSPF is stable. This has shown considerable improvement over cases where no such techniques were applied.

It is expected that the same solution gives similar results, for different sources of flaps such as link flaps and adjacency flaps, for other link state routing protocols such as IS-IS [1], and even for general distributed systems that distribute and synchronize information.

This chapter has described a contribution to improving the availability of a routing system, which is a representative network element. In the next chapter, we continue with our overall strategy to improve the reliability of the Internet by addressing multipath systems.

Chapter 6

MARA: Maximum Alternative Routing Algorithm

6.1 Overview

In this chapter, a family of new multipath routing algorithms called Maximum Alternative Routing Algorithm (MARA) are introduced. The multipath routing algorithms are used to improve the network availability in the following chapter.

Routing systems in the Internet play a very important role, deciding the paths that will be used for all communication sessions. The path that the routing system calculates determines factors such as the reachability of the destination, the available amount of bandwidth, and the communication delay.

The most essential task of routing systems is to calculate routes that do not include routing loops and are consistent on all routers. The Internet is designed as a hop-by-hop network so that routers in the network need not maintain per-session communication state, improving scalability in performance, number of nodes, and number of sessions. Since all routers in the Internet forward packets autonomously, they must independently make consistent decisions about the path to a destination in order to avoid routing loops. Configuring consistent routes in a network for a destination without routing loops is synonymous with constructing a Directed Acyclic Graph (DAG) for the destination. The presence of directed edges in the DAG that eventually sink to the destination without any cycles indicates that valid routes have been calculated in each router.

To a first approximation, routing in the Internet is single-path routing. Its goal is to find the shortest path by assigning each edge a cost and making the routing decision so that, for each source and destination pair of nodes, the sum of the edge costs in the path is the minimum. This principle can be seen in all commonly used routing protocols, including RIP [54], OSPF [59], IS-IS [1], and BGP [82].

Compared to the single-path routing commonly used in the Internet, enabling the use of multiple paths simultaneously, called *multipath routing*, can improve availability and transport capacity. However, in the current shortest-path routing, each router usually selects a single shortest path to a destination. Thus, the routes to the destination form a sink tree to the destination (called a Shortest Path Tree, or SPT), where multiple paths in the network can rarely be used. Although the existing IP routing architecture allows multipath routes where the path from the current node to the destination branches to multiple nexthops, multipaths in the shortest path routing require the costs to the destination on these paths to be equal, which rarely happens. The rare multipath routes in the shortest path routing are called Equal Cost Multi Paths (ECMPs) [59]. Note that both SPT and its ECMP extension (no longer precisely a tree) are still a kind of DAG.

Multipath routing can contribute to the availability of the communication network. Although it is commonly believed that the Internet routing systems avoid failures, in the real world, complex problems that cannot be detected by routing systems occur, such as hardware malfunctions and software bugs in routers [84, 64]. In theory, the rich connectivity of the Internet should exhibit high reliability. In practice, the Internet routing protocols reduce the richer graph to a non-redundant tree. The non-redundant tree is vulnerable to link failures, which induce route recomputations and degrade communication performance and connectivity for the duration of the recomputation, and to problems in the forwarding plane of routers, which may remain undetected for extended periods and may ultimately require manual intervention. Advance calculation of multipaths along with a path switching mechanism that does not depend on the failure detection of a particular routing system can contribute to the availability of a communication system. One example of such a path switching mechanism is Deflection [109], where the end host can request a change of the communication path by changing the packet tag embedded in IP packets.

Multipath routing may also contribute to the transport capacity of network. In the case of shortest-path routing, the maximum available bandwidth between a source and a destination is limited to that of the shortest path. This means that the transport capacity of a network is unreasonably constrained, despite the possible availability of alternative paths with additional bandwidth. Multipath routing can alleviate this problem by splitting traffic to multipaths to balance the network load. Although MPLS [83] may be considered to be a multipath routing mechanism, this work also considers networks that do not use MPLS.

This chapter proposes a family of novel multipath route calculation algorithms, called Maximum Alternative Routing Algorithm (MARA), that construct a DAG that includes all edges in the network graph structure, in order to provide the maximum number of alternative paths to all nodes in the network. This is the first study of such DAGs for Internet routing, to the best of our knowledge. Three new graph problems, *all-to-one maximum connectivity routing problem*, *all-to-one max-flow routing problem*, and *all-to-one maxi-*

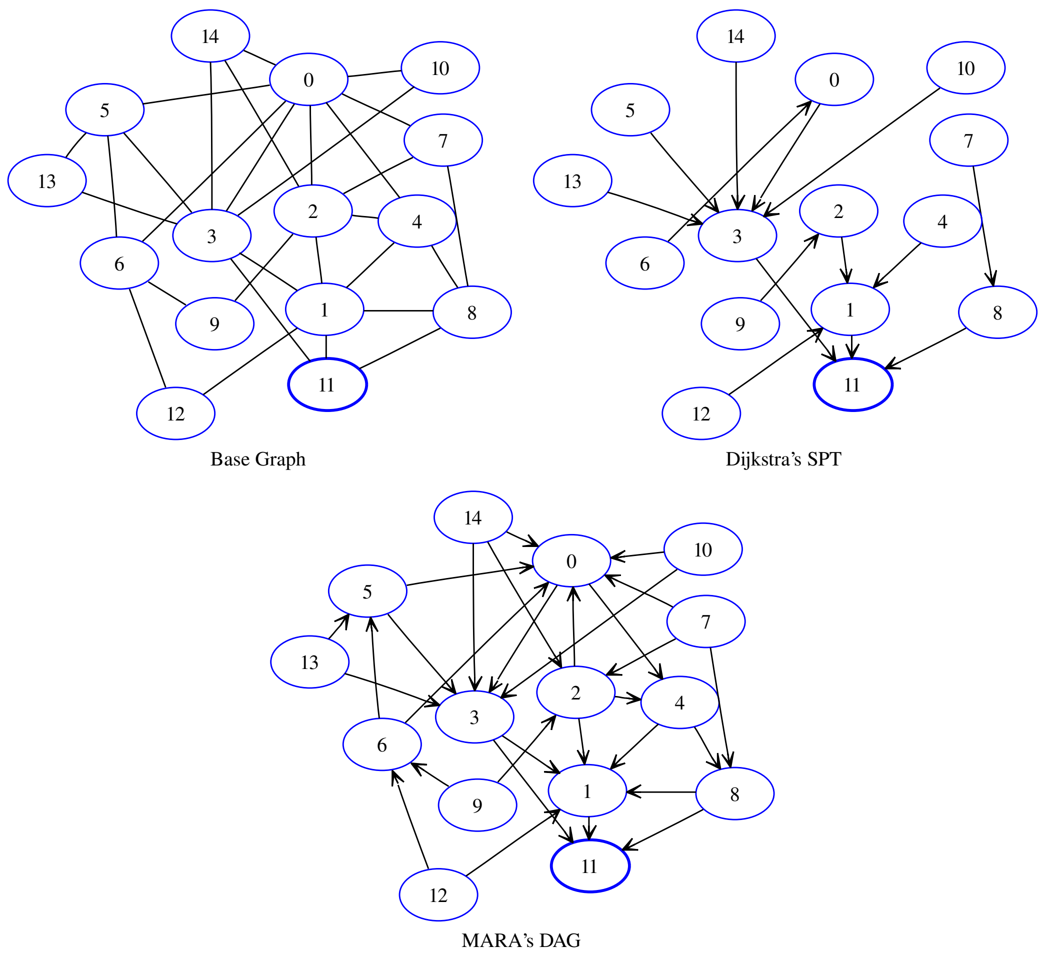

Figure 6.1: Example of routes calculated by SPT and by MARA.

mum shortest path alternatives problem are introduced, to describe the objectives of the algorithms in this chapter. The new routing algorithms that solve the problems optimally, called MARA-MC, MARA-MMMF, and MARA-SPE, are defined as applications of an existing algorithm called Maximum Adjacency Ordering (MA ordering [61, 62]) algorithm to the Internet routing.

The example of routes calculated by existing Dijkstra's algorithm and by MARA is illustrated in Figure 6.1. The network graph is shown in "Base Graph" (upper-left part in the figure), and the routes for the destination node 11 are illustrated as directions on the edges. Dijkstra's algorithm tends to calculate a single shortest-path route, as shown in "Dijkstra's SPT" (upper-right part in the figure). MARA calculates multipath routes that include all edges in the graph, as shown in "MARA's DAG" (bottom in the figure). The MARA's DAG, in contrast to the Dijkstra's SPT, provides redundancy in the routing level which is expected to contribute to failure recovery, and provides options to each node which route to use, to reach the destination.

We proposed Load Balancing Routing Algorithm (LBRA) [68]. The work of LBRA inspired us to define the all-to-one maximum flow routing problem, which in turn led to the optimal utilization of MA ordering algorithm.

The main contributions of this chapter are: (a) the introduction of the new graph routing problems; (b) the proposal of new multipath routing algorithms that apply MA ordering algorithm to the Internet routing; (c) the optimality proofs; and (d) the analysis of the algorithms on ISP topologies down from the current Internet.

The rest of this chapter is organized as follows. Section 6.2 presents the related work of multipath routing algorithms. Section 6.3 defines the problems. Section 6.4 describes the algorithms and optimality proofs. Section 6.5 presents analysis of the algorithms as an evaluation. Section 6.6 gives the summary of this chapter.

6.2 Related Work

Maximum flow algorithms such as Dinits' algorithm [25] consider the max-flow for a source-destination node pair in the graph, while the all-to-one max-flow routing problem considers the max-flow among all nodes to the destination.

Some multipath route calculation algorithms have been proposed in the past. Multipath routing methods proposed in the past are based on, and are extensions of, the shortest path routing. Hence they require the routing metric setting in advance. MPDA [104] is a link state routing algorithm which distributes only partial topology information. MDVA [106] is a distance vector routing algorithm that uses diffusing computation [24]. MPATH [105] is another distance vector routing algorithm that distributes predecessor node information of paths. MPDA, MDVA and MPATH calculate multipath routes that are loop-free at every instance, using the Loop Free Invariant (LFI) condition on the routing metrics. MARA does not consider the routing metrics specified by network administrators, and calculates routes directly on graph attributes such as link bandwidth.

FIR [52] computes per network interface routing tables by executing the Shortest Path First (SPF) calculations separately for each of its neighbors in order to route around the failure. In [109], Yang and Wetherall proposed Deflection, which extends the LFI condition by utilizing the identity of the previous hop to produce an increased number of nexthops. FIR and Deflection are multipath routing methods for the purpose of failure avoidance. They provide backtracking paths that transit the same node twice, while the method in this chapter provides simple hop-by-hop multipath routes without backtrackings.

6.3 Problem Definitions

An undirected graph $G = (V, E)$ and a destination node $t \in V$ in the graph are given. For the destination, it is required to calculate a routing direction on every edge in the graph. The directions on edges indicate routes for the hop-by-hop routing from every node to the destination on the graph. To avoid routing loops, the resulting directed graph should not have any directed cycles; i.e., the goal is to find a directed acyclic graph (DAG) on the input graph. The DAG is sometimes referred to as a *routing graph*.

A number of DAGs is allowed for a hop-by-hop network to be used as a routing graph. In the rest of this section, three problems are defined for the calculation of a DAG that achieves particular objectives. Note that a DAG signifies routing directions for a destination node. Thus multiple DAGs must be calculated to produce routes for all source-destination pairs in the network.

Remark This chapter takes undirected graphs as input, only for ease of explanation. The algorithms in this chapter support directed graphs, as is; that is, for a given directed graph and a destination node, a DAG that achieves the particular objective should be found, without any modification required. The only difference of directed input graphs from undirected input graphs is that the explanation "a DAG which includes all edges in the graph" no longer conforms. For example, when an undirected graph is translated to a directed graph, an edge in the undirected graph is expressed as a pair of edges with opposite directions in the directed graph. Then either one of the edge would not be used in the DAG calculated by the algorithms in this chapter.

6.3.1 All-to-one Maximum Connectivity Routing Problem

The first objective is to calculate a DAG with robustness. To improve the robustness of the routing graph in the network, it is required to maximize the edge connectivity from all nodes to all destinations. The robustness of reaching from all nodes to the destination in the network is determined by the minimum connectivity in the routing graph. By maximizing the minimum connectivity among all nodes to the destination node, an ideal routing that achieves the highest robustness among all nodes to one destination node can be obtained. The problem of deciding the directions of edges in order to maximize the minimum connectivity among all nodes to a destination is called the *all-to-one maximum connectivity routing problem*. By calculating such routing for every destination, the whole routing that achieves the most robustness for all source-destination pairs can be obtained.

A formal description of the all-to-one maximum connectivity routing problem is as follows. An undirected graph $G = (V, E)$ is given, as the graph structure of the network on which the routes are to be calculated. The input graph is assumed to be connected and may have multiple edges between a pair of nodes, i.e., multigraph. A node $t \in V$ is also given as the destination.

A possible solution (a routing graph, DAG) is expressed by an orientation of edges (in other words, a set of directions of all edges). For each orientation p of edges, $k_p(v,t)$ denotes the edge connectivity from v to t in the DAG determined by the orientation p. ($k_p(t,t) = +\infty$ is assumed for every orientation p.) The all-to-one maximum connectivity routing problem is to find an orientation of edges in the given graph that maximizes the minimum connectivity among all nodes to the destination t under the condition that the resulting directed graph is acyclic. The problem is formulated as follows.

$$\begin{array}{ll} \text{Maximize} & \min_{v \in V} k_p(v,t) \\ \text{subject to} & p \text{ is an acyclic orientation.} \end{array}$$

An algorithm to solve this problem and an optimality proof for the algorithm are given in Section 6.4.2.

6.3.2 All-to-one Max-flow Routing Problem

The next objective is to support as high traffic load as possible. A typical routing algorithm uses only the shortest path to support traffic load between each source-destination pair. By using roundabout paths, it becomes possible to support traffic load with excess amount for the shortest path. The traffic which can be supported from a node to the destination in the network is determined by the max-flow of the routing graph. Since the minimum of max-flows among all nodes to the destination node (called the bottleneck) limits the supported range of the traffic demands, it is important to increase the minimum max-flow. By maximizing the minimum max-flow to the destination among all nodes, an ideal routing that achieves the best support for the excessive amount of traffic to the destination can be obtained. This problem is called the *all-to-one max-flow routing problem.* By calculating such routing for every destination, the routing that achieves the best support for all source-destination node pairs can be obtained.

A formal description of the all-to-one max-flow routing problem is as follows. An undirected graph with link capacity $G = (V, E, cap)$ is given, where $cap(v,u)$ is the capacity of the edge (v,u) if $(v,u) \in E$, otherwise 0 ($cap(v,u) > 0$ if and only if $(v,u) \in E$). A node $t \in V$ is also given as the destination.

A possible solution (a routing graph, DAG) is expressed by an orientation of edges. For each orientation p of edges, $f_p(v,t)$ denotes the amount of max-flow from v to t in the DAG determined by the orientation p. ($f_p(t,t) = +\infty$ is assumed for every orientation p.) The all-to-one max-flow routing problem is to find an orientation of edges in the given graph that maximizes the minimum max-flow among all nodes to the destination t under the condition that the resulting directed graph is acyclic. The problem is formulated as follows.

$$\begin{array}{ll} \text{Maximize} & \min_{v \in V} f_p(v,t) \\ \text{subject to} & p \text{ is an acyclic orientation.} \end{array}$$

An algorithm to solve this problem and an optimality proof for the algorithm are given in Section 6.4.3.

6.3.3 All-to-one Maximum Shortest Path Alternatives Problem

Most existing Internet routers employ shortest-path routing. When creating a new routing graph, it may be preferable to construct a DAG such that it includes the shortest path tree (SPT). If a new routing is consistent and is a superset of the shortest path routing, new routers and shortest path routers can coexist in the network simultaneously. Routing algorithms proposed in the past (e.g., FIR [52], N-hub [17], and Deflection [109]) extend or utilize the shortest path routing.

The objective of this section is to calculate a DAG that is consistent and is a superset of the shortest path routing, in order to obtain higher robustness in routing graphs. The problem of deciding the directions of edges to maximize the minimum connectivity among all nodes to a destination under the constraint of including SPT and the graph extended by ECMP, is called the *all-to-one maximum shortest path alternatives problem.* It is also possible to consider a problem of calculating a DAG that extends the shortest path routing, with the best traffic load support.

A formal description of the all-to-one maximum shortest path alternatives problem is as follows. An undirected graph with link costs $G = (V, E, cost)$ is given, where $cost(v,u) > 0$ gives the cost of the edge (v,u) if $(v,u) \in E$, otherwise $+\infty$ (the path with cost $+\infty$ is considered as unreachable). A node $t \in V$ is also given as the destination.

For an orientation p of edges, $k_p(v,t)$ denotes the edge connectivity from v to t in the DAG determined by the orientation p. ($k_p(t,t) = +\infty$ is assumed for every orientation p.) The all-to-one maximum shortest path alternatives problem is to find an orientation of edges in the given graph that maximizes the minimum connectivity among all nodes to the destination t under the condition that the resulting directed graph is acyclic and the orientation is a superset of the orientation in the shortest path tree (denoted by p_{spt}). The problem is formulated as follows.

$$\begin{array}{ll} \text{Maximize} & \min_{v \in V} k_p(v,t) \\ \text{subject to} & p \supseteq p_{spt}, \\ & p \text{ is an acyclic orientation.} \end{array}$$

An algorithm to solve this problem and an optimality proof for the algorithm are given in Section 6.4.4.

6.4 Algorithms

In this section, algorithms to solve the problems denoted in Section 6.3 are described. Instead of deciding the direction of each edge independently, a permutation of nodes is determined by the algorithms (i.e., nodes are labeled from 1 to n). A permutation of nodes sets the direction of each edge from the higher-labeled node to the lower-labeled node. It is known that constructing a DAG on an undirected graph is equivalent to deciding a topological order of nodes [2].

First, the MA ordering proposed by Nagamochi and Ibaraki [61] is reviewed in Section 6.4.1. In Sections 6.4.2, 6.4.3, and 6.4.4, new algorithms based on the MA ordering are proposed in order to solve the problems introduced in Sections 6.3.1, 6.3.2, and 6.3.3, respectively.

6.4.1 MA Ordering

Let $G = (V, E)$ be an undirected graph that has n nodes and m edges. An ordering $v_1, v_2, \ldots, v_n$ of nodes is called an MA ordering if an arbitrary node s is chosen as a starting point v_1, and after choosing the first i nodes as $v_1, v_2, \ldots, v_i$, the $(i+1)$-st node v_{i+1} is chosen from the remaining nodes v that have the largest number of edges between v and $\{v_1, \ldots, v_i\}$. It is known that MA ordering is useful for various problems on graphs, such as identifying a minimum cut between two nodes and solving the edge-connectivity augmentation problem [62]. An algorithm to compute an MA ordering is given in Algorithm 1, where $d(v, S)$ denotes the number of edges between a node v and a set of nodes S (i.e., the number of edges from v to one of the nodes in S).

For a capacitated, undirected graph $G = (V, E, cap)$, an ordering similar to the MA ordering is also defined. In this case, instead of choosing a node $v \in T$ with the largest $d(v, S)$ in Line 5, a node v with the largest $\sum_{u \in S} cap(v, u)$ is chosen.

By using an appropriate data structure such as Fibonacci heap [32], an MA ordering for an undirected graph $G = (V, E)$ and a node $s \in V$ can be obtained in $O(m + n)$ time for the case without link capacity and $O(m + n \log n)$ time for the case with link capacity [61].

6.4.2 MARA-MC

Now an algorithm that solves the all-to-one maximum connectivity routing problem optimally is proposed. The algorithm is called MARA-MC, where MC stands for maximum connectivity. MARA-MC is very simple: An MA ordering for an undirected graph $G = (V, E)$

Algorithm 1 MA ordering algorithm.

```
1: procedure MA ORDERING($G = (V,E), s \in V$)
2:     $v_1 \leftarrow s,\ S = \{s\},\ T = V \setminus \{s\}$
3:     $i \leftarrow 2$
4:     while $i \leq |V|$ do
5:         choose a node $v \in T$ with the largest $d(v,S)$
6:         $v_i \leftarrow v,\ S = S \cup \{v\},\ T = T \setminus \{v\}$
7:         $i \leftarrow i+1$
8:     end while
9:     output ordering $(v_1, v_2, \ldots, v_n)$ of nodes
10: end procedure
```

and an destination node $t \in V$ is computed using the destination t as the initial node s in Algorithm 1. Then the direction of each edge is set from the higher-labeled node to the lower-labeled node. The direction on an edge is interpreted as a route to the destination, with the head of the directed edge being the nexthop of the route. The MA ordering calculates routes from all nodes to the destination t. In order to determine routes among all source-destination node pairs in the network, the MA ordering algorithm must be executed for each destination in the network separately, i.e., n times. Hence MARA-MC runs in $O(m+n)$ time for a destination; $O(mn+n^2)$ time for all source-destination pairs. A node must calculate routes for all source-destination pairs to find the routes from the node to all destinations in the network.

Below, an optimality proof for MARA-MC is given. Let us define the bottleneck node v in an orientation p as the lowest-labeled node with the minimum $k_p(v,t)$, where an orientation p of edges is given with a permutation of nodes; i.e., nodes are labeled from 1 to n and each edge is headed to the lower-labeled node. First, in the following lemma, it is shown that the minimum cut between v and the destination t is always the neighboring cut of the bottleneck node v.

Lemma 6.4.1. *Let v be the lowest-labeled node with the minimum $k_p(v,t)$. Then, $k_p(v,t) = d(v,V')$ holds, where V' is the set of nodes which have lower-labels than v.*

Proof. By using a relationship between the cut and the connectivity on a graph, $k_p(u',t) \leq d(u',U)$ holds for every node $u' \neq t$, where U is a set of nodes having lower-labels than u'. In other words, the connectivity from u' to t, i.e., $k_p(u',t)$, cannot exceed the number of edges in its neighboring cut, $d(u',U)$.

The equality in Lemma 6.4.1 is proven by contradiction. Suppose that $k_p(v,t) < d(v,V')$ holds. The max-flow min-cut theorem on a directed graph [22] implies that there exists a directed cut X (i.e., a partition of nodes) whose size is equal to the connectivity from v to t,

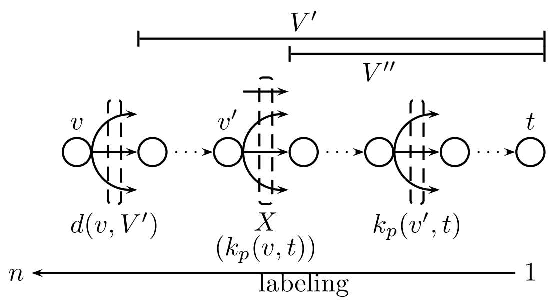

Figure 6.2: The relation between v, V', v', V'', X and t in Lemma 6.4.1.

i.e., $k_p(v,t)$. Let us see Figure 6.2 as an example. Figure 6.2 shows the cut X in the center, and v' is assumed to be the lowest-labeled node such that v and v' belong to the same subset of nodes partitioned by X. Then, the following inequalities hold:

$$k_p(v,t) \geq d(v',V'') \geq k_p(v',t), \tag{6.1}$$

where V'' is the set of nodes which have lower-labels than v' (V'' is also shown in the upper right in Figure 6.2). This contradicts the assumption that v is the lowest-labeled node with the minimum connectivity to t. □

Theorem 6.4.2. *MARA-MC solves the all-to-one maximum connectivity routing problem optimally.*

Proof. Suppose that the MA ordering algorithm using the destination node t as an initial node gives a label i to node v_i for every node in V (i.e., the destination node has label 1, $t = v_1$). Call the orientation of edges from the higher-labeled node to the lower-labeled node "*ma*," and let g_{ma} be the objective value for this orientation, i.e., the minimum connectivity among all nodes to the destination t . $V_i = \{v_1, v_2, \ldots, v_{i-1}\}$ denotes the set of nodes which have smaller labels than v_i. By the definition of $d(v,V')$, $d(v,B) \leq d(v,A)$ if $B \subseteq A$ holds.

Let v_k be the lowest-labeled node whose connectivity $k_{ma}(v_k,t)$ to the destination node t in "*ma*" is the smallest. By using Lemma 6.4.1, the following equalities hold:

$$g_{ma} = k_{ma}(v_k,t) = d(v_k,V_k). \tag{6.2}$$

From a property of the MA ordering algorithm (in Line 5 of Algorithm 1), $d(v_i,V_k) \leq d(v_k,V_k)$ holds for $i = k+1, k+2, \ldots, n$, because v_k is chosen earlier than v_i.

Assume that the optimal ordering, whose orientation is denoted by "*opt*," is better than the MA ordering (i.e., $g_{opt} > g_{ma}$ holds). Let v_l be the node with the smallest label in *opt* among a set of nodes $\{v_k, v_{k+1}, \ldots, v_n\}$. In other words, v_l is the node which is to the left

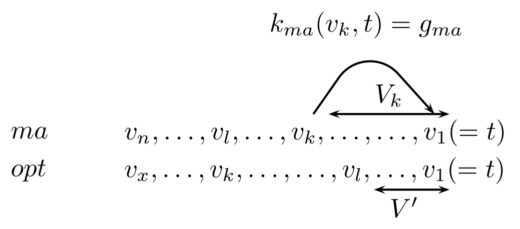
$k_{ma}(v_k,t)=g_{ma}$
V_k
ma $v_n,\dots,v_l,\dots,v_k,\dots,\dots,v_1(=t)$
opt $v_x,\dots,v_k,\dots,\dots,v_l,\dots,v_1(=t)$
V'

Figure 6.3: The optimality proof of MARA-MC.

of v_k in "*ma*", but is the rightmost node in "*opt*" among the nodes $\{v_k,v_{k+1},\dots,v_n\}$. This is illustrated in Figure 6.3. Then, the following equalities and inequalities hold:

$$g_{opt} \le k_{opt}(v_l,t) \le d(v_l,V') \le d(v_l,V_k) \le d(v_k,V_k) = g_{ma}, \tag{6.3}$$

where V' is the set of nodes having smaller labels than v_l in *opt* (shown in bottom right of Figure 6.3). This contradicts the assumption that there exists a better orientation *opt* with $g_{opt} > g_{ma}$. □

6.4.3 MARA-MMMF

The algorithm that solves the all-to-one max-flow routing problem optimally is called MARA-MMMF, where MMMF stands for maximizing the minimum max-flow. MARA-MMMF computes an MA ordering for an undirected graph with link capacity $G=(V,E,cap)$ using $\sum_{u\in S} cap(v,u)$ instead of $d(v,S)$ to consider the max-flow rather than the connectivity. The difference between MARA-MC and MARA-MMMF is only the input (the graph contains link capacity) and the preference on the node (larger max-flow is preferred over larger connectivity). MARA-MMMF runs in $O(m+n\log n)$ time for a destination node, and in $O(mn+n^2\log n)$ time for all node pairs.

The optimality proof for MARA-MMMF is very similar to that of MARA-MC. The optimality of MARA-MMMF can be obtained by transforming the proof of MARA-MC where $d(v,S)$ and $k_p(v,t)$ are substituted by $\sum_{u\in S} cap(v,u)$ and $f_p(v,t)$, respectively.

6.4.4 MARA-SPE

In this section, an algorithm that solves the all-to-one maximum shortest path alternatives problem is proposed. The algorithm is called MARA-SPE, where SPE stands for shortest path extension. MARA-SPE consists of two stages: (1) a shortest path tree (or a partial order of nodes in general case) on the input graph $G=(V,E,cost)$ is computed using the destination node t as the root of the tree, (2) a routing graph from all nodes to the destination is calculated under the condition that the resulting directed graph is acyclic and all edges in the shortest path tree have directions from leaves to the root (i.e., toward the destination node t) in the DAG.

In the first stage, MARA-SPE computes a shortest path tree by using Dijkstra's algorithm. This is done in $O(m+n\log n)$ time. In the second stage, MARA-SPE computes an ordering of n nodes with an algorithm similar to the MA ordering algorithm. In Line 5 of Algorithm 1, a node v with the largest $d(v,S)$ was chosen among all $v \in T$ for an MA ordering. Instead of this condition, a node v with the largest $d(v,S)$ was chosen among all nodes $v \in T'$, where $T' \subseteq T$ is a set of nodes whose ancestors in the SPT are already chosen in Line 5 (i.e., all the ancestors of the node v in the SPT belong to S). It is possible to choose a node $v \in T'$ with the largest $d(v,S)$ in $O(\log n)$ time with a heap [89].

Note that a very similar algorithm can be applied to the following problem: Find an orientation of edges in the given graph that maximizes the minimum max-flow among all nodes to the destination t under the condition that the resulting directed graph is acyclic and the orientation is a superset of the orientation in the shortest path tree (denoted by p_{spt}). MARA-SPE (and the very similar algorithm) run in $O(m+n\log n)$ time for a destination node, and in $O(mn+n^2\log n)$ time for all node pairs.

At the end of this section, let us see an optimality proof for MARA-SPE. First, MARA-SPE always output a permutation of n nodes; that is, the size of node set T' is at least 1 for any $i \leq |V|$ in Line 5 since the shortest path tree (or its extensions) must not have any directed cycles. It is also clear that the resulting DAG includes all the edges in the shortest path tree with those correct directions (this property also comes from Line 5). The optimality on the objective value can be obtained very similar to that of MARA-MC.

6.5 Evaluation

MARA-MC is evaluated for the number and the length of paths, the computational complexity, and the computation time. Since the primary objective of MARA are to calculate as many alternative paths as possible, the number of paths is the most important metric for the purpose of this work. Advantage in the number of paths of MARA-MC over another multipath calculation algorithm, LFI, is shown.

For computation time, an implementation of the MARA-MC with a (basic) heap (i.e., it runs in $O((m+n)\log n)$ time) is evaluated in this section. Even though there exists a faster $O(m+n)$ implementation for all-to-one maximum connectivity routing problem as mentioned earlier, this version of MARA-MC gives, at least, an indicative estimate whether MARA, especially MARA-MC and MARA-MMMF, are practical. Evaluation of other related algorithms, such as the faster algorithm of all-to-one maximum connectivity routing problem and MARA-SPE are left as future work.

Table 6.1: Average and standard deviation of the number, the average length, and the maximum length of paths. The average length increases with multipath algorithms due to the active use of secondary, longer paths.

		#paths		avg path len		max path len	
AS num	algorithm	avg	stddev	avg	stddev	avg	stddev
AS1221	Dijkstra	1.37047	0.62119	5.79252	1.77890	5.93393	1.86591
	LFI	3.36870	3.90845	6.03323	1.84686	6.34677	2.01281
	MARA-MC	26.27146	31.46643	8.24450	2.98650	10.60143	4.30103
AS1755	Dijkstra	2.07190	1.69354	6.07565	2.13137	6.23442	2.22470
	LFI	16.58821	44.47381	6.57977	2.21208	7.31221	2.66163
	MARA-MC	872.52753	2335.68504	11.70987	4.53271	16.44426	6.79061
AS3257	Dijkstra	2.14138	1.97148	6.82304	2.74271	7.17437	2.94942
	LFI	607.56405	2999.14628	7.98135	3.09047	9.65547	4.33032
	MARA-MC	10176.30097	37125.45047	12.40258	4.54230	18.45062	7.50249
AS3967	Dijkstra	1.88704	1.69201	5.87894	2.15526	6.08406	2.36544
	LFI	75.98442	232.45061	7.02206	3.00882	8.35832	4.05909
	MARA-MC	42.08098	66.93976	8.61961	3.30758	10.95245	4.63834

6.5.1 Number and Length of Paths

MARA-MC is evaluated by comparing the number and length of the computed paths on topologies collected using Rocketfuel [92]. The number of paths is calculated by enumerating all of the paths on the DAG, for each source-destination pair. Results for four ASes, AS1221, AS1755, AS3257, and AS3967, are shown in Table 6.1. Since the enumeration of paths takes time exponential in the number of edges, the evaluation of two larger ISPs, AS1239 and AS6461, are omitted here, but will be included in results below. The Dijkstra algorithm used in the evaluation calculates ECMP properly.

The average and standard deviation of the number, the average length, and the maximum length of paths among source-destination pairs are shown in Table 6.1. The table shows that on most topologies, MARA-MC outperforms LFI in the number of calculated multipaths. In AS3967, LFI outperforms MARA-MC in the average number of paths. This is because LFI computed a large number of paths to a small set of source-destination pairs (more than 1000 paths to 1.31% fraction of source-destination pairs). The average and maximum length of paths tend to become larger when more multipaths are calculated. The use of multipaths implicitly presumes the use of longer roundabout paths.

The CCDF of the number, the average length (in nodes), and the maximum length of paths are shown in Figures 6.4 and 6.5. The figures show that MARA-MC consistently calculates a large number of multipaths for most source-destination pairs, while the ability of LFI to find multipaths is heavily dependent on the topology. Dijkstra, which calculates only equal-cost multipaths, finds few for most source-destination pairs. For example, in

Table 6.2: Computational complexities of routing algorithms. l denotes the number of neighboring nodes for a node, m the number of edges in the graph, and n the number of nodes.

	At each node	Overall
Bellman-Ford	$O(mn)$	$O(mn^2)$
Dijkstra	$O(m+n\log n)$	$O(mn+n^2\log n)$
LFI & Deflection-1	$O(lm+ln\log n)$	$O(mn+n^2\log n)$
Deflection-2	$O(lm+ln\log n+l^2n)$	$O(mn+n^2\log n+m^2)$
Deflection-3	$O(lm+ln\log n+l^2n)$	$O(m^2n+mn^2\log n)$
MARA-MC	$O(mn+n^2)$	$O(mn+n^2)$
MARA-MMMF	$O(mn+n^2\log n)$	$O(mn+n^2\log n)$

AS1221, MARA calculates more than 15 multipaths for 52.65% of source-destination pairs, while LFI does for only 2.17% . The average path length and the maximum path length of LFI and Dijkstra are roughly the same for both AS1221 and AS1755. MARA-MC exhibits longer path length. For example in AS1221, the fraction of source-destination pairs that has more than 8 node path length is 14.89% for Dijkstra, and 16.5% for LFI, while it is 55.96% for MARA-MC.

In AS3967, the average number of paths by LFI outperformed MARA-MC (Table 6.1). However, the distribution of number of paths in AS3967 still exhibits the preferable characteristic of MARA. In the low range, MARA-MC provides the same paths to larger fraction of source-destination pairs than LFI. For example, LFI provides more than 15 paths to 38.43% of source-destination pairs, while MARA-MC provides to large fraction, which is 53.03%. Providing a small number of multipaths to larger part of the network is preferred over providing a large number of multipaths to a small part of the network, for failure avoidance purpose.

6.5.2 Computational Complexity

Table 6.2 gives the computational complexities for several algorithms. The middle column is the run time for the algorithm as run at *each* node (router) in the network; the right-hand column is the *network-wide total* computation necessary for all nodes to calculate the routes from all nodes to all destinations. l denotes the number of neighboring nodes for a node, m the number of edges in the graph, and n the number of nodes.

Bellman-Ford [37] is the routing algorithm utilized in Distance Vector routing algorithms such as RIP. For each node, it calculates n routes in at most m steps, and globally it calculates this n times.

The complexity of Dijkstra, used in OSPF and IS-IS, is similar to that of MA Ordering, in that it is $O(m+n\log n)$ by appropriate use of Fibonacci heap. Since Dijkstra calculate all routes from a source node to all destinations, globally it must be calculated n times (one for each source node).

Deflection-1, Deflection-2 and Deflection-3 indicate Rule-1, Rule-2, Rule-3 of Deflection, respectively. Deflection-2 and Deflection-3 are important theoretical mileposts for comparison, but cannot practically be deployed, because they calculate routes with routing loops that require special treatment in packet forwarding and routing tables. They are described here for comparison purpose, only.

Deflection-1 is equivalent to LFI, which is used in MPDA, MDVA, and MPATH. For a node, LFI and Deflection-1 require the Dijkstra calculation for each neighboring node, and comparisons of the neighbor's cost with the node's cost for each destination, which is $O(ln)$. The complexity of LFI and Deflection-1 is globally n times Dijkstra plus m times cost comparisons for each destination, which is $O(mn)$.

Deflection-2 compares the costs of previous hop in the packet forwarding path (called *previous node*), with that of (possible) successor nodes in the path. Hence, at each node it requires Dijkstra l times and cost comparison $\binom{l}{2}$ times for each destination, leading to the complexity $O(lm + ln\log n + l^2n)$. Globally it requires Dijkstra n times in addition to the cost comparisons for all neighboring pairs of nodes (m times, for *downhill*), and for all neighboring pairs of edges ($\binom{m}{2}$ times in the worst case, for *two-hop*). Hence the global complexity is $O(mn + n^2\log n + m^2)$.

Deflection-3 compares a successor node's cost on a modified graph with one edge removed, the node's cost on another modified graph depending on the previous node, and the previous node's cost on the original graph, to calculate routes from a node to a destination. This approach means that Deflection-3 requires Dijkstra on $3l$ different graphs, and cost comparison $\binom{l}{2}$ times for a destination in a node. This leads to the complexity $O(lm + ln\log n + l^2n)$ at each node. Globally Deflection-3 requires Dijkstra for each one edge removed from the graph, for each destination (which is mn times). The cost comparisons are performed for all neighboring pairs of nodes (m times, for *downhill*) and for all neighboring pairs of edges ($\binom{m}{2}$ times in the worst case, for *two-hop*). Hence the global complexity is $O(m^2n + mn^2\log n)$.

The bottom line of the table shows the complexity of two of MARA algorithms. Unlike the above algorithms, MARA does not prune the graph; the connectivity from nodes *not* on the shortest path can still contribute to multipaths, so all edges in the graph are retained. Conversely, the calculation is the same at every node or router; it is, in theory, possible to share results of the calculation among multiple nodes if desired.

Of course, the $O(\cdot)$ notation, by definition, hides a constant factor and transient terms in the execution time. Those terms can be of critical importance when considering whether to deploy a system in the real world. The next subsection addresses these concerns.

6.5.3 Computation Time

The computation times of three of the routing algorithms are shown in Figure 6.6. The algorithms are implemented in C and executed on a Linux PC with Intel® Pentium® D 3.20GHz CPU and 4GB memory. The algorithms use the more practical priority queue, i.e., a heap, rather than a Fibonacci heap, hence the complexities differ somewhat from those shown in Section 6.5.2. Our implementation of Dijkstra is $O((m+n)\log n)$, LFI $O(l(m+n)\log n)$, and MARA-MC $O(n(m+n)\log n)$. The computation time to calculate routes to all destinations in the network is measured 1000 times for each source node. The calculation naturally requires a different amount of time for each node in the network, as reflected in the the CDF of the computation time, as shown in Figure 6.6. The time is measured by `gettimeofday()` in a user-mode software application, hence the operating system's process scheduling may be negatively affecting the variability of the results. Allocation and deallocation of the necessary memory is included in the computation time. Additionally, MARA is implemented as labeling algorithm, while LFI and Dijkstra are implemented so that many memory allocations are necessary. This means that LFI and Dijkstra can be implemented more faster. However, MARA's feasibility remains the same even when the other algorithms are implemented more efficiently.

The results show that the computation time of MARA-MC is reasonable. In all of the ASes examined, the execution times are tightly clustered and are only a small fraction of a second. In most cases, MARA-MC is faster than LFI. For example, 94.25% fraction of executions of MARA-MC for AS1755 terminate within 12 milliseconds, while only 36.29% terminate in the case of LFI.

AS1221 and AS6461 include a disconnected component in the graph. Since Dijkstra finds only the routes for connected component from the source, the computations for those nodes terminate quickly. MARA does not determine whether the destination node is connected from the computation node during the course of the computation, and hence calculate all routes for nodes in disconnected component, which eventually fail to find routes to other parts of the graph. This suggests that the MARA should be used after checking the basic connectivity of the graph.

6.6 Summary

Migrating the Internet from essentially single-path routing to multipath routing can potentially improve the fault resilience of the network, raise the aggregate bandwidth available between two nodes, and increase the utilization of otherwise idle resources. One step toward the realization of this goal is the development of algorithms for finding the multipath routes on the network graph.

Toward this goal, three new multipath routing problems are introduced. The problem of finding a DAG that includes all edges, and that maximizes the minimum connectivity among the nodes to a destination, is defined as the *all-to-one maximum connectivity routing problem.* The similar problem of maximizing the minimum max-flow we call the *all-to-one max-flow routing problem.* Finally, the *all-to-one maximum shortest path alternatives problem* is the problem of maximizing the minimum connectivity while satisfying the condition of being consistent with the existing shortest path routing algorithms.

A family of three optimal algorithms to solve these problems are developed, which we call MARA-MC, MARA-MMMF, and MARA-SPE. This family is called MARA, for *Maximum Alternative Routing Algorithms.* All three algorithms calculate a DAG that includes all of the edges in the graph. These routing algorithms compute many multipaths, which may be used in the Internet to improve failure avoidance. MARA-MC is evaluated on the number and the length of paths and on the algorithm's computational time, using topologies inferred from real, large networks. The results showed that MARA-MC calculates a significant number of multipaths; for the several autonomous systems (ASes) evaluated, the lowest average number of paths found was more than twenty-five. The computation time on a modern Intel processor is sub-one second, verifying its feasibility in practice.

In this chapter, routing algorithms that calculate many multipaths with objectives of maximizing connectivity and max-flow are developed. The following chapter presents how to use these routing algorithms to improve network availability.

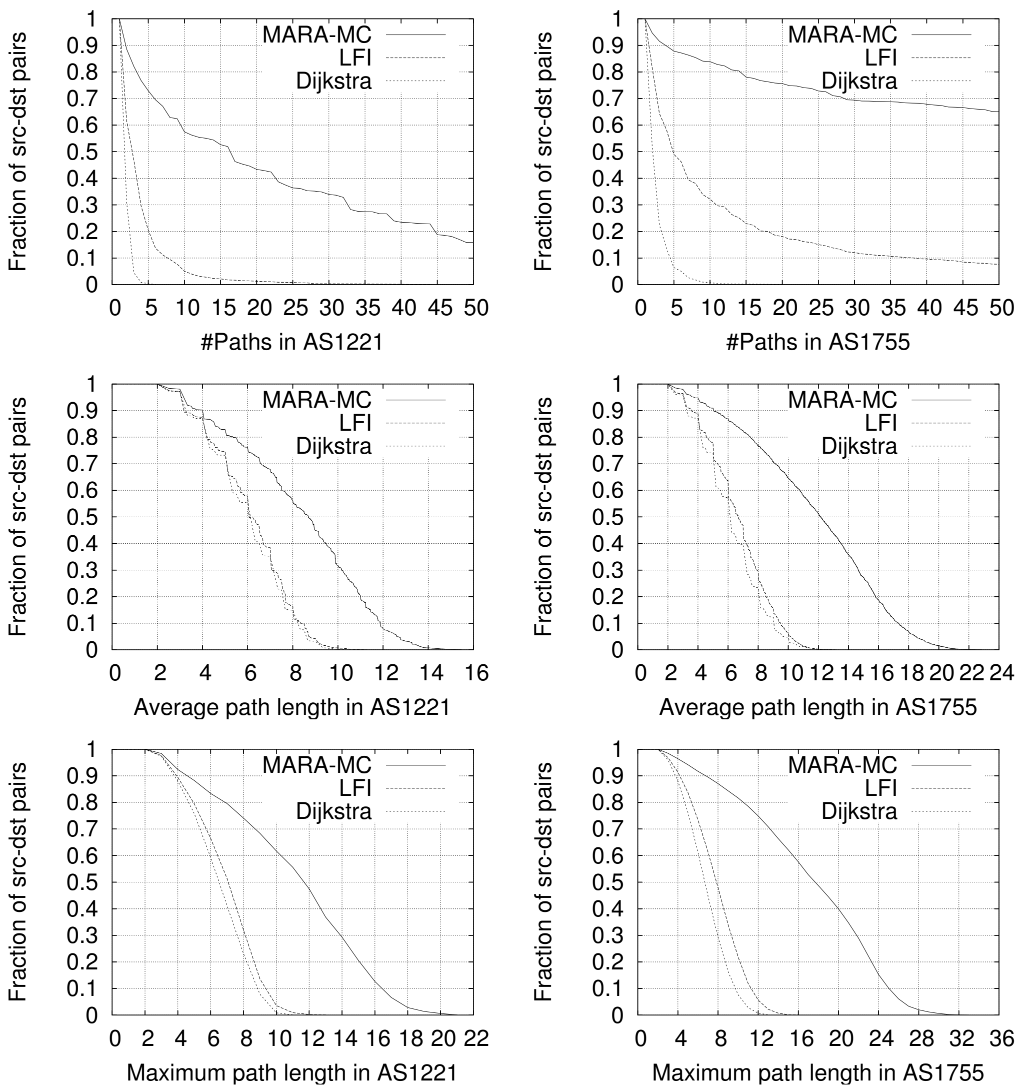

Figure 6.4: Comparison on distributions on the number, the average, and the maximum length of paths for AS1221 and AS1755.

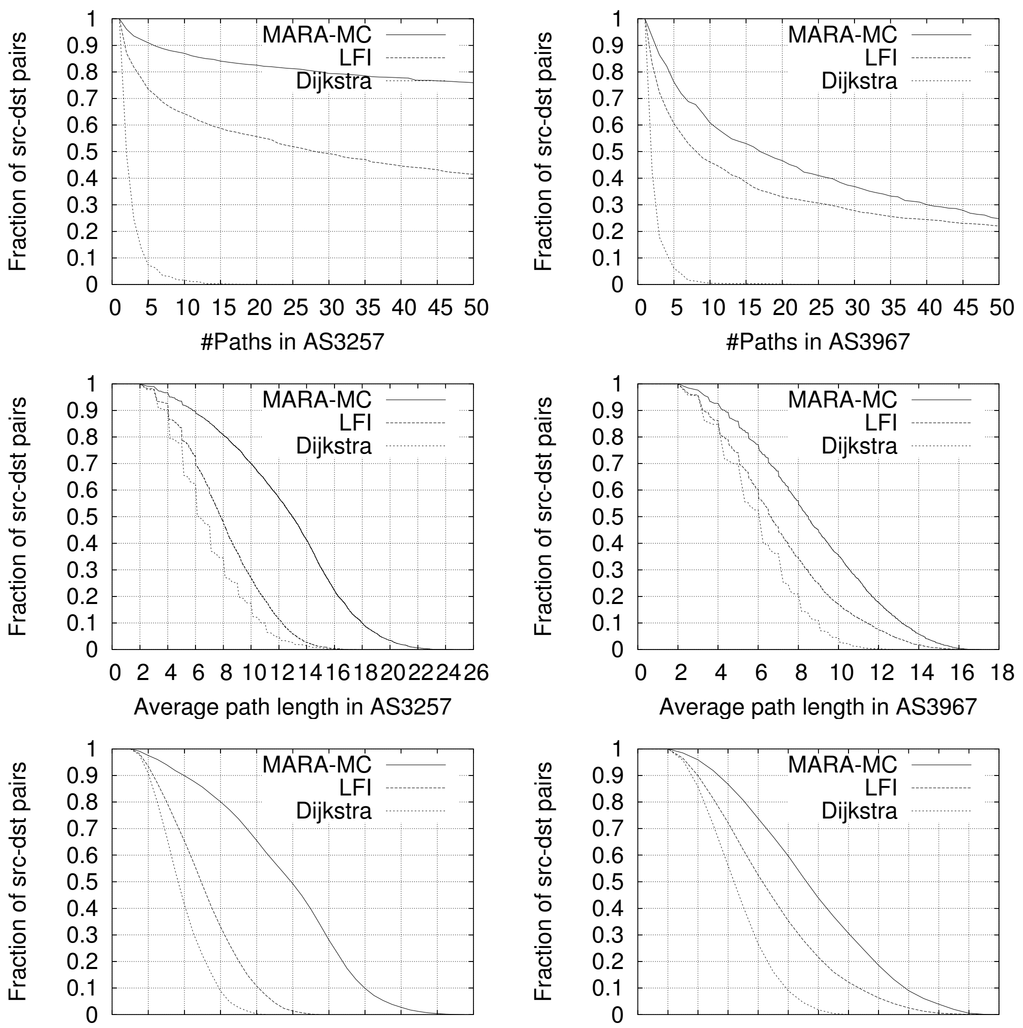

Figure 6.5: Comparison on distributions on the number, the average, and the maximum length of paths for AS3257 and AS3967.

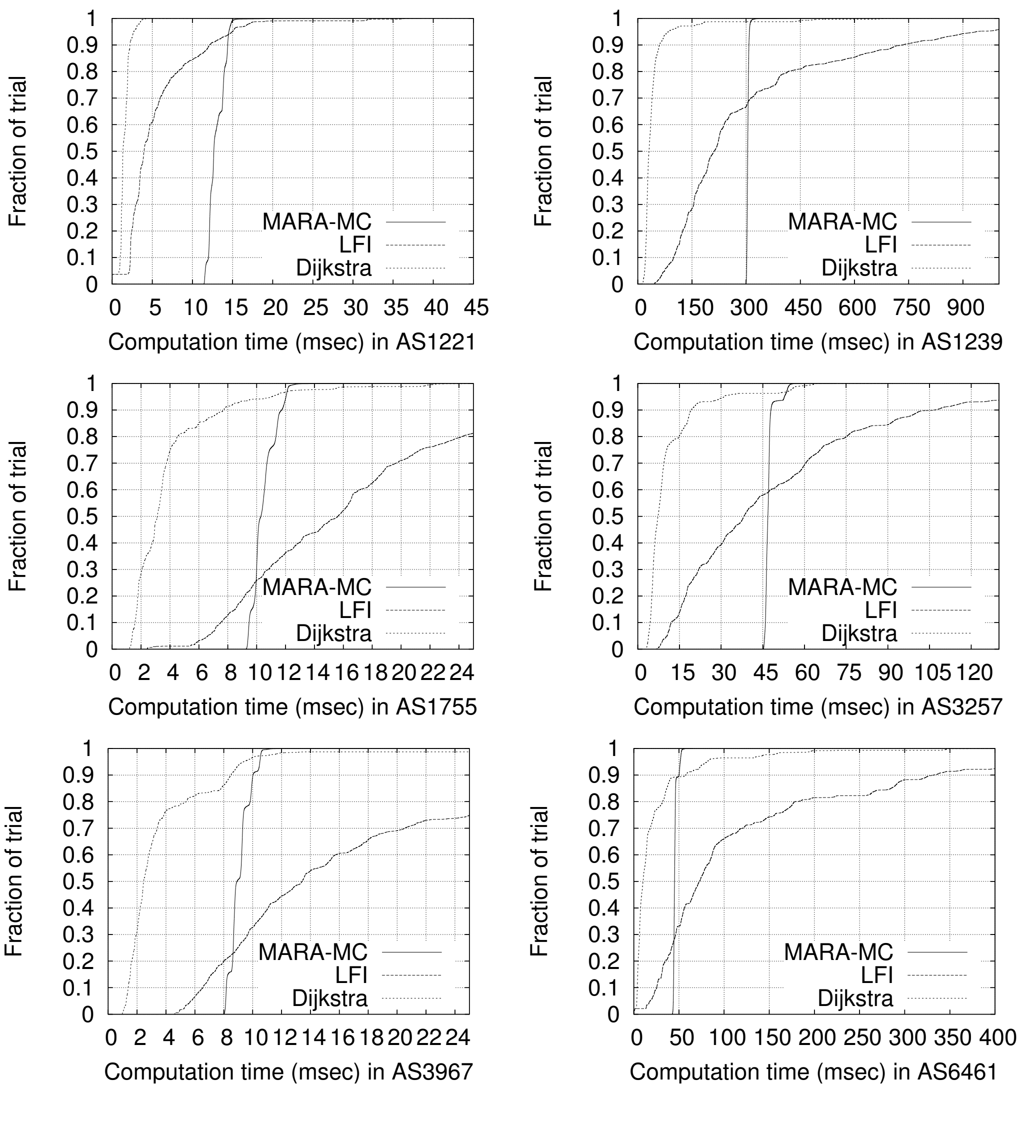

Figure 6.6: Computation time of algorithms.

Chapter 7

Drouting Architecture: A Multipath Routing Architecture

7.1 Overview

The Internet communications are disrupted by failures of routers, circuit failures, fibre optic cable cuts, misconfigured routers, software bugs in routers, and many other reasons. In general, communication reachability is recovered through the recomputation of alternative routes. Routing systems have the responsibility to detect failures and to recompute alternative routes.

An important problem arises here that there are many cases where the routing system cannot detect the failures, due to its unlimited variety of causes. For example, malfunctions caused by either hardware or software bugs in the forwarding plane cannot be detected through OSPF Hellos [59] or BGP Keepalives [82]. In case of a failure that cannot be detected by the routing system, it may take a long time (e.g. a few hours) for the network to recover, as the network administrator informed of the failure by users examines and recovers it manually. Although the length of the failure duration depends on the individual networks, past studies showed that a small fraction of failures can last more than an hour. In a study at an ISP [50], approximately 20% of failures lasted more than two hours. It is shown in [40, 56] that about 10% of real failures lasted more than 20 minutes, and approximately 5% lasted more than one hour, in the Sprint backbone.

Many methods have been proposed to minimize service downtime, such as multipath routing, IP restoration, and path protection (see Section 7.2 for related work). Recently a method called Deflection has been proposed [109]. Deflection constructs a multipath routing on the existing shortest-path machinery and utilizes those roundabout paths when the packet tag is changed by the source host, as explained in Section 7.3.

In this chapter a novel routing architecture called Drouting is proposed. Drouting simplifies and generalizes Deflection and improves the ability to route around failures. The Drouting architecture consists of two components. First is the multipath route calculation component. It computes multipath routes by constructing Directed Acyclic Graphs (DAGs) that include all links in the network. DAGs that include all links in the network have not been studied for the purpose of the Internet routing. The second component of Drouting is the tag forwarding framework. The IP packet forwarding procedure is slightly changed to select the actual nexthop from the precomputed multipath routes based on the packet tag. A packet tag corresponds to a network path deterministically without any state management on routers. The tag forwarding component enables end hosts to change a route dynamically, which can be based solely on user preferences.

Drouting architecture is evaluated by comparing it to Deflection using simulations. The comparisons are on: (1) the number of nexthops, (2) the number of paths, (3) the length of paths and (4) the probability of recovery from failures by changing the packet tag. Drouting exhibits similar results with Deflection in the first three points, while in terms of the failure recovery probability, Drouting outperforms Deflection on most topologies.

The rest of this chapter is organized as follows. Related works are given in Section 7.2. In Section 7.3 the Deflection is reviewed as a competitor. In Section 7.4 the Drouting architecture and how it works are described. In Section 7.5 the evaluation results and its analysis are presented. This chapter is concluded in Section 7.7.

7.2 Related Work

RON [5], Detour [86] and PlanetLab [74] utilize overlay networks for the purpose of failure recovery. MPDA [104], MDVA [106] and MPATH [105] are multipath routing algorithms that calculate non-shortest multipaths. They use a condition for the routing metric called the Loop Free Invariant (LFI). FIR [52] computes routing tables per network interface using the previous hop information in the routing calculation in order to improve network availability when transient failures occur. IPFRR [7] explores calculating loop-free alternate nexthops using a routing metric condition similar to LFI. Deflection [109] extends LFI (which is equivalent with what they call Rule-1) and calculates more multipaths depending on the previous hop. It also proposed a packet tag system for end-systems to "deflect" the path upon failures.

Until Deflection, little has been discussed regarding an actual architecture to utilize multipaths for the purpose of failure recovery.

7.3 Deflection Architecture

In this section the Deflection architecture [109] is reviewed to understand for later comparison with our architecture. Deflection is akin to multipath routing schemes. It calculates multipaths using the specific conditions of routing metrics and the identity of previous hop in the forwarding of the packet. The nexthop set that forms a loop-free path is called the "deflection set". There are three conditions of routing metrics for a neighbor to decide whether or not the neighbor can be put in the deflection set. The first, called Rule-1, is equivalent to LFI [104, 106, 105]. Rule-1 specifies that the neighbor having a lower cost to the destination can be used as a nexthop of a valid route where no routing loop is guaranteed. Rule-1 calculates only a small deflection set for a router, and the use of a Rule-1 deflection set does not achieve sufficient failure recovery capability. Hence, the second condition, Rule-2, is made to extend Rule-1 by using the identity of the previous hop. Rule-2 specifies (in addition to Rule-1) that the neighbor having a lower cost than the previous hop can be used as a nexthop. Rule-2 calculates many members in the deflection set, and provides sufficient failure recovery capability, but it can result in a backtracking path that transit a same node twice. For this reason the path of Rule-2 tends to become longer. To avoid this problem, Rule-3 is introduced with more complex conditions to prevent immediate backtracks in a path. Rule-3 calculates the shortest path costs on modified graphs where the link from the previous hop to the calculating node, or from the calculating node to the neighbor, is removed. Then it compares those costs to decide whether the neighbor can be a nexthop.

Among the three rules, Rule-3 is the best solution and is the major competitor with our method, since Rule-1 calculates too few nexthops and Rule-2 calculates unrealistic network paths, which backtrack and transit a same node twice. For this reason, Rule-2 is not considered as a viable competitor in this chapter.

Deflection allows the end host to voluntarily try a different path through the network. In Deflection, packet tags are used to switch paths within the multipath set. When the end host detects (or suspects) a failure in the network that the routing system has not yet corrected, it can switch the packet tag and possibly bypass the failure until the network heals itself. If, after changing the packet tag, packets still do not reach their destination, the end host can try another packet tag. Up to ten packet tags are tested; the first five tags to be tested are 1 through 5, then the next five are chosen randomly from the range $[6, 1023]$.

In order for routers not to synchronize with other routers in terms of the forwarding direction for the same packet tag, Deflection proposes to prepare for each router a prime number p from the first primes (e.g. the first 10) greater than or equal to k (where k is the size of the deflection set). Then the router uses p to form $n = (t \bmod p) \bmod k$, where t is the packet tag, and the packet is forwarded to the n-th member in the deflection set. These rules are followed in later simulations.

There are paths that cannot be used in the network, because of the following two reasons. First, two or more routers may choose the same prime number for p. If this condition occurs, the routers will synchronize in terms of selection of the n-th nexthop in the deflection set, resulting in the possibility of non-used combinations of nexthops. Second, because the number of possible tags is 1024, at most 1024 paths can be used for a source-destination pair. If the network has more than 1024 paths in a source-destination pair, those paths greater than the number 1024 cannot be used.

Assuming m and n are the number of links and nodes in the graph respectively, and k is the number of neighbors for a router, complexity for Deflection Rule-3 in a router is that of $3k$ times* Dijkstra's SPF [23], hence $O(km + kn\log n)$. Calculation of deflection sets for all routers in the network can be done by one Dijkstra for each router plus one Dijkstra for each case of one link removed in the network graph. Hence, the complexity is $O((m+n)(m+n\log n))$.

7.4 Drouting Architecture

7.4.1 Overview of Drouting Architecture

Drouting is similar to Deflection in that both calculate multipath routes and switch a path using a packet tag. Compared to Deflection, Drouting generalizes the specification of the packet tag. In Drouting the packet tag is a random value, while in Deflection choosing specific packet tags is proposed. Another difference is the method of calculating the multipath. Deflection calculates multipath routes by using the identity of the previous hop (incoming hop) of a packet and relations between routing metrics. Drouting does not depend on routing metrics and calculates simple hop-by-hop multipath routes without using the identity of the previous hop.

The route calculation component of Drouting computes multipath routes by constructing Directed Acyclic Graphs (DAGs) that include all links in the network. DAGs that include all links in the network have not been studied for the purpose of the Internet routing.

The tag forwarding component enables end hosts to dynamically change a path based on user preferences. A packet tag is assigned to a network path deterministically without having to maintain any states on routers. The packet tag is randomly chosen. A source host changes its packet tag only when it desires to use another network path. In order to avoid packet reordering and degradation of TCP performance, source hosts are assumed to assign the same packet tag for all the packets in one TCP session.

*Deflection calculates the shortest paths on the graph without the link to the nexthop and with the nexthop as a root, on the graph without the link from previous hop and with the router itself as a root, and on the graph with the previous hop as a root.

In the beginning, the source host initiates a communication using packet tag 0. The packet tag 0 is treated as special packet tag that instructs the network to use the default shortest path. The source host is assumed to detect problems on the communication path in some way such as a fixed timer for packet losses or dynamic bandwidth estimation (for methods to detect the bandwidth problems, refer to [45]. This problem is beyond the scope of this chapter). Once the source host detects a problem, it randomly chooses a new packet tag, such as `0x159bf`. The new packet tag is expected to be assigned to a new communication path, which may stochastically recover from the problem.

Introducing a special packet tag 0 provides the separation of default routing plane and backup routing plane. Although Drouting can be used also for the default routing plane, this chapter focuses on the use only in the backup routing plane, in order to compare the failure recovery property with Deflection fairly. When Drouting is used also for default routing plane, a failure case (a combination of source-destination pair and a failure node, upon which comparisons are based) for Drouting and Deflection would be different, because Drouting's default path may traverse the failure node depending on the value of the first packet tag. (See Section 7.5.1 for the details of the failure recovery simulation.) Thus the use of Drouting also in default plane makes the evaluation of failure recovery property by comparison harder, and studying the use of Drouting in default routing plane is left as future work.

In our architecture, a source node can neither predict nor specify in advance which network path will be assigned for its packets. A network path is only randomly assigned to a packet as a result of forwarding the packet with the particular packet tag. However, a source node can specify the same network path for multiple communication sessions to the same destination, by using the same packet tag.

Changing the packet tag enables recovery from a long failure even if the routing system fails to detect the network failure. If a failure occurs in the network and the routing system can detect the failure, the routing system will automatically recompute network routes, hence altering network paths to recover from the failure, the same as in the existing Internet. A source host can use an alternative path by changing the packet tag, regardless of whether or not the routing system detects (and hence will route around) the failure. By changing the packet tag, the network path which the packet will take may or may not be changed, depending on the randomly generated new packet tag.

Our architecture can be used to minimize perceived network downtime as end hosts do not have to wait for the routing system to finish route recomputations. Furthermore, end hosts can possibly address QoS performance problems by changing the packet tag when the performance of the current path deteriorates. For example, congested links and/or overloaded routers can be avoided to improve QoS performance.

Figure 7.1 shows an overview of the Drouting architecture. Components modified from

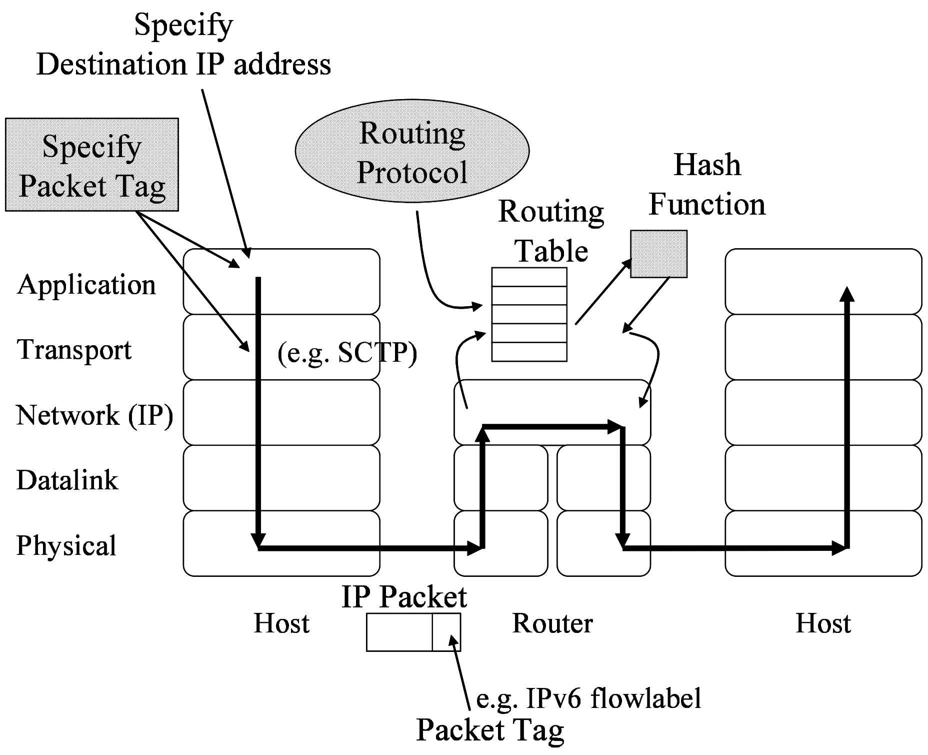

Figure 7.1: Overview of the Drouting architecture.

the existing routing architecture are shaded. Notice that Drouting is an intuitive extension of the basic multipath routing. Only the routing protocols and packet forwarding engine are slightly changed without adding a significant new complication. The packet tag is assumed to be stored in the IPv6 flowlabel [81], hence no packet format modification is needed. Because the length of the IPv6 flowlabel field is 20 bits, the packet tag has the range [`1-0xfffff`] in this chapter. Changing the packet tag is the task of either application software or a transport protocol such as SCTP [96]. The routing algorithm in the routing protocol is discussed in Section 7.4.2. The actual design of the routing protocol is beyond the scope of this chapter. The packet forwarding framework is discussed in Section 7.4.3.

7.4.2 Multipath Route Calculation

To guarantee loop-free packet forwarding and that each packet will eventually reach its destination, routes must be calculated such that routes for a given destination construct a DAG that sinks to the destination. Each route corresponds to a directed link in the DAG. By following the nexthops of the routes hop-by-hop in the DAG, a packet will eventually reach its destination without any routing loop.

In the Drouting architecture a method to construct DAGs that include all of the links

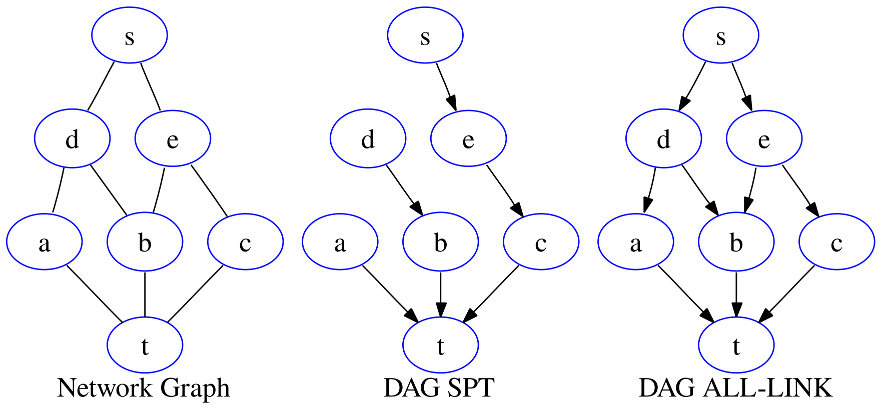

Figure 7.2: A network graph, the shortest path tree (SPT) as it would be calculated today, and the DAG that includes all links.

in the network graph is proposed, in order to provide the maximum number of alternative paths in hop-by-hop routing. The difference between the DAGs utilized by the existing routing systems and the DAGs that include all links in the network graph is illustrated in Figure 7.2. Shortest Path Tree (SPT), as shown in the middle of the figure, has been used for routing on the example network graph shown in the left of the figure. Use of the DAG that include all links (ALL-LINK), as shown in the right of the figure, is proposed to improve the failure recovery capability.

A DAG that includes all links can be constructed by topological ordering [2] in which nodes are labeled from 1 to n and the links are directed from the higher-labeled node to the lower-labeled node. To construct a DAG that sinks to the destination, labeling is started from the destination node.

We proposed constructing DAGs using Maximum Adjacency (MA) ordering [62] to make DAGs that have the maximum number of paths to the destination. Drouting employs this method, and the failure recovery property is investigated through simulations for the first time in this chapter. The formal problem definition and proof for the multipath calculation method are given in Chapter 6.

The complexity of the MA ordering to calculate a DAG is $O(m+n\log n)$. Each router must calculate all DAGs for each destination. Hence, the complexity of Drouting for both a router and the overall network is $O(mn+n^2\log n)$.

7.4.3 Tag Forwarding

An end host randomly chooses the tag for Drouting in the range [1,`0xfffff`]. The network path to be used by a traffic flow will be selected stochastically from the combination of the nexthops in all intermediate routers.

In Drouting, packet tags are random values. Based on this random value, and through the random switching of a nexthop in the nexthop set in the routers, a stochastic selection of a network path for a packet tag is achieved.

Each intermediate router performs a routing table lookup by using the IP destination address as the key (as in the existing Internet). The resulted routing table entry contains multiple nexthops to the destination. The router selects the nexthop randomly by the packet tag. The assignment of a packet tag to a nexthop is performed deterministically, such that: (1) intermediate routers do not need to maintain the state of each traffic flow (i.e., TCP session or IP source-destination pair), and (2) randomly chosen packet tags yield a network path randomly.

A perfect hash function $F_x : U \mapsto U$ (U is the set represents the tag space) which is different for each router v_x is employed. The nexthop selection is $n = F_x(t) \bmod k$ (where t is the packet tag, k is the number of nexthops in the routing table entry, and the selected nexthop is the n-th nexthop in the routing table entry). The F_x can be implemented as follows. First, prepare an table $\mathrm{tab}[|U|]$ and initialize $\mathrm{tab}[t] = t$ for each tag $t \in U$. Next, swap $\mathrm{tab}[t]$ with $\mathrm{tab}[i]$ of a random index i for each tag t.

7.4.4 Differences from Deflection

A summary of the differences of Drouting from Deflection is described here.

- Deflection limits the number of tags to 1024, hence the number of paths Deflection provides are up to 1024. Drouting limits the number of paths to `0xfffff`-1 (subtracting special default tag 0).
- The prime number chosen by each router in Deflection may synchronize. This leads to unused combinations of nexthops, i.e., unused paths. Drouting does not have this problem.
- The methods used to calculate multipath routes are different. The method of Deflection considers relations between routing metrics to calculate multipaths, while Drouting calculates the multipaths with maximum connectivity to the destination. Complexity for a router is $O(km + kn\log n)$ in Deflection, and $O(mn + n^2 \log n)$ in Drouting. This difference comes from the fact that, if a Deflection router has k neighbors, it calculates the routing tree k times, once assuming each neighbor is the previous hop (removed from the graph to avoid immediate backtracking). Hence the complexity of Deflection is parameterized by k (the number of neighbors). In contrast, Drouting decides the routing graph (basically not tree, because it includes all links), for each destination. Hence the complexity of Drouting is parameterized by n (the number of destination nodes). The difference of complexity is hence that of k (more precisely, $3k$)

and n, and can be deemed to be negligible. Moreover, Deflection requires the memory space to hold the results of each of $3k$ Dijkstra runs to finish the calculation process for a router. In Drouting it is simpler because the result for a destination is independent from the result for other destinations, and the memory used for the calculation of different destination can be released. In summary, the complexities for a router to execute routing calculation in a network in both methods are equivalent to a constant multiple of Dijkstra, and hence they are both realistic and reasonable.

7.5 Evaluation

Drouting architecture is evaluated by comparison with Deflection on several topologies using simulations. The topologies were obtained from Rocketfuel [92]. The size of each network graph and the description are shown in Table 7.1. Telstra, Sprint, Ebone, Tiscali, Exodus and Abovenet are ISP topologies inferred by Rocketfuel.

Table 7.1: The network graphs for simulations of Drouting architecture.

Name	#nodes	#links	description
Telstra	108	153	AS1221
Sprint	315	972	AS1239
Ebone	87	166	AS1755
Tiscali	161	328	AS3257
Exodus	79	147	AS3967
Abovenet	141	374	AS6461

Four systems are compared on these topologies: Deflect-1, Deflect-2, Deflect-3 and Drouting. The notations Deflect-1, Deflect-2, Deflect-3 indicate Deflection routing Rule-1, Rule-2, Rule-3 in [109] respectively. Deflect-3 is the major competitor, as mentioned in Section 7.3.

The comparisons were done in four aspects: (1) the probability of success to recover from failures (i.e., the failure recovery probability), (2) the number of nexthops, (3) the number of paths, and (4) the length of paths. Improving the failure recovery probability is the primary purpose of the four systems, and thus evaluated first. Comparisons in the number of nexthops, the number of paths, and the length of paths are performed to reveal overheads and side-effects of these technologies. The simulation method and the interpretation of path comparisons are given in Sections 7.5.1 and 7.5.2, respectively.

The results of simulations and analysis on these topologies were largely the same. Thus, only the result on the network topology of Telstra is given in this chapter. The results of simulations and analysis on other topologies are given in Appendix B.

7.5.1 Failure Recovery Simulation Method

The failure recovery simulations were done as follows. First, a node was randomly chosen to fail. Then, for every source-destination pair, it was checked whether the failure node was on the default shortest path calculated by the Dijkstra calculation. For each source-destination node pair that included the failure node in the default shortest path (this is called "a failure case"), a tag was chosen to see if the retry using the new tag avoids the failed node in forwarding the packet. If the retry did not avoid the failed node, another new tag was tested. At most ten tags were tested. For each failure case, the number of tags tested before the failed node was successfully avoided is recorded. This procedure was executed for each method, Deflection and Drouting, with the same source-destination pair and failure node. 1000 random failure cases are tested for both Deflection and Drouting, and recorded the fraction of failure cases in which the failure was successfully avoided within x tag changes (retries).

Additionally, similar failure recovery simulation with 10 failed nodes are performed. A case where one of the failed nodes affect the default shortest path is called "a 10 failures case". The simulation is called the "10 failures recovery simulation". The single failure version is called the "single failure recovery simulation" in contrast to the 10 failures recovery simulation.

Notice that the assumption here is that Drouting is used as a backup routing plane, where the routing plane is constructed separately from the default routing plane. This is just like Deflection, and is done to compare with Deflection fairly. This assumption makes Figure 7.1 incorrect in a precise sense, because after the assumption a router has two routing tables, one for default routing and the other for Drouting.

As for Drouting, the simulation results shown here utilize the retry packet tag range of [0-`0xfffff`] instead of [1-`0xfffff`], which is not precisely the same as the expected deployment as backup routing plane in real network. However, the difference is negligible; it is expected that just 1 smaller range of packet tag does not influence the major results of the failure recovery property.

7.5.2 Interpretation of path comparison

The fraction of source-destination pairs having only one nexthop indicates the number of single point of failures in the network, and hence provide the degree of weakness of the network. A smaller number of routers that have only one nexthop to the destination is preferred. However, too many nexthops in a router indicate the inefficient use of memory, as it would not be necessary for avoiding a small number of network faults (we assume that the number of network faults that occur simultaneousely is usually small, such as at most 10). The distribution of the number of nexthops among routers where the majority of routers have a small number of nexthops equally is preferred, over the distribution where a small number of routers have huge nexthops and others have a few nexthops. This is because it is desired to provide multipaths for greater number of source-destination pairs, for failure recovery purpose.

The comparison on the number of paths shows the simplest degree of failure avoidance possibility. Additionally, a large number of paths with a small number of nexthops means the efficient calculation of multipath routes. The number of paths in Deflection was counted by examining tags from 0 to 1023 and counting the unique paths, as was done in [109]. The number of paths in Drouting is counted by enumerating all possible paths. The numbers of paths counted by this way represent the actual paths that can be used in each routing method.

The length of paths indicate performance degradation as the side-effect of improving the failure recovery probability, because the longer the path, the more the communication delay. Further, use of the longer path means inefficient utilization of link bandwidth, as the same traffic will consume additional bandwidth in longer paths. Although the shorter length of the paths are preferred, the use of multipath naturally involves the use of longer, roundabout paths, and thus the length of paths tends to become longer when more number of multipaths are calculated.

7.5.3 Simulation and Analysis on Telstra (AS1221)

The network topology of Telstra is shown in Figure 7.3. The isolated component of the graph is due to the topology data provided by the Rocketfuel project.

The result of the single failure recovery simulation on the Telstra topology is shown in Figure 7.4, and the result of the 10 failures recovery simulation is shown in Figure 7.5. The fraction of failure cases in the single failure recovery simulation in which the first retry avoided the failure was 25.1% in Drouting, while it was only 14.9% in Deflect-1, 14.7% in Deflect-2, 11.8% in Deflect-3. Up to 10 retries, Drouting has successfully avoided 55.2% failure cases, while Deflect-1 has 26.6%, Deflect-2 has 39.7%, and Deflect-3 has 41.2%. The failure avoidance probability of Drouting is significantly larger than Deflection's. Even in the 10 failures case (Figure 7.5) Drouting recovers 40.4% of the failure cases up to 10 retries, outperforming Deflect-3's 30.6%.

The distribution of the number of nexthops among routers are shown as Complementary Cumulative Distribution Function (CCDF) in Figures 7.6. In the figure, Deflect-2 has the largest number of nexthops. Then, briefly, Deflect-3, Deflect-1 and Drouting follow in the order. Deflect-1 and Drouting have largely similar numbers of nexthop distribution. In

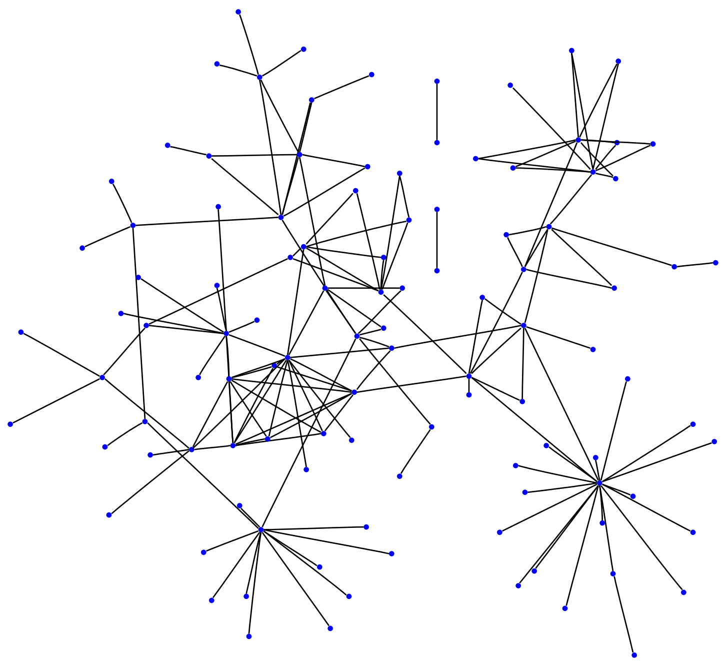

Figure 7.3: The Telestra's network topology.

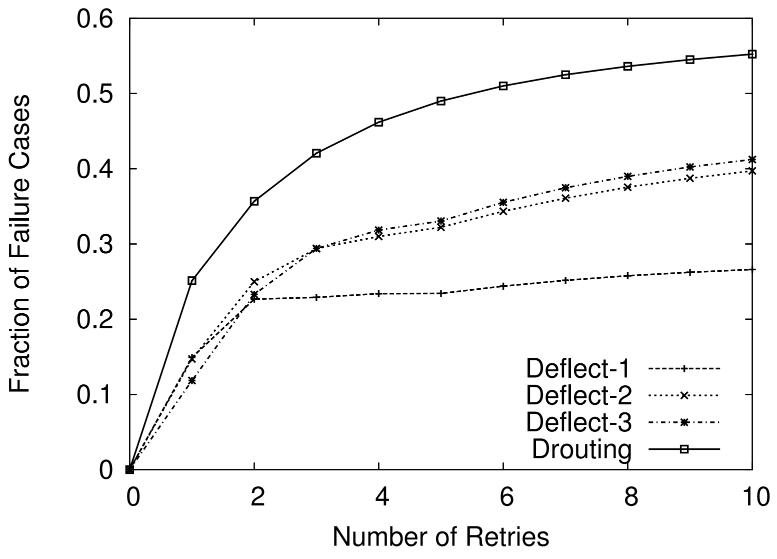

Figure 7.4: The fraction of failure cases, successfully recovered by different systems on the Telstra topology.

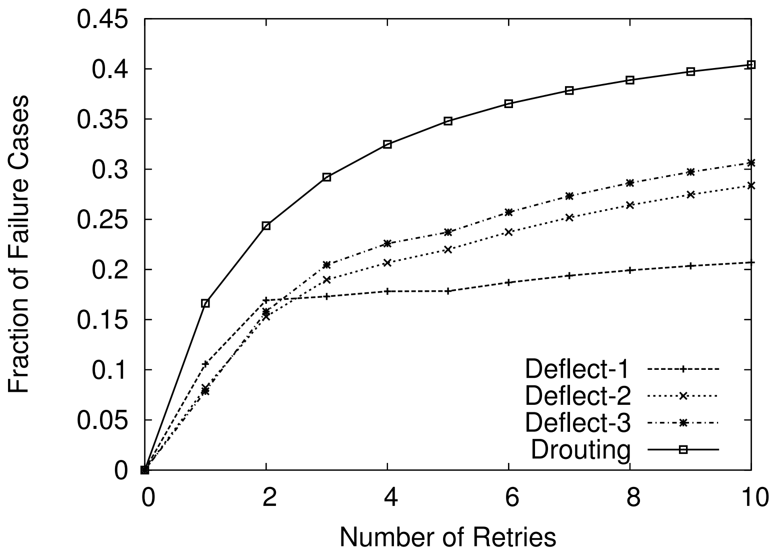

Figure 7.5: The fraction of 10 failures cases, successfully recovered by different systems on the Telstra topology.

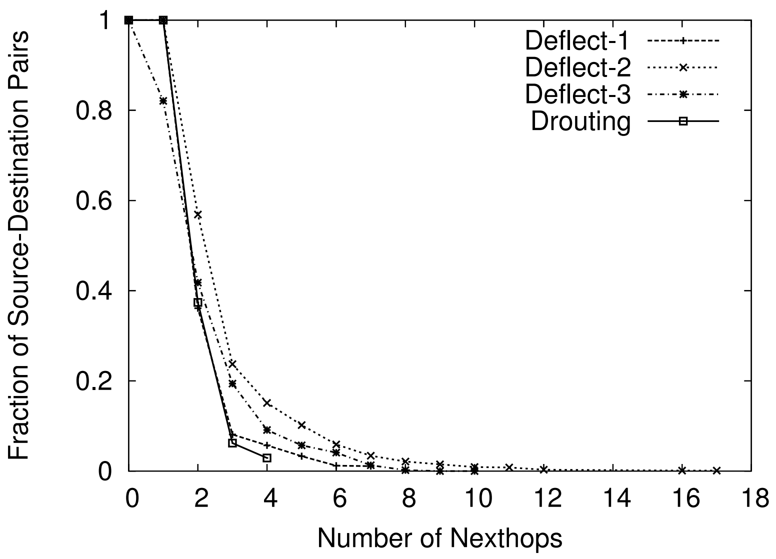

Figure 7.6: The fraction of source-destination pairs on the Telstra topology, having more than x nexthops.

Deflect-1, 36.1% of the source-destination pairs have 2 or more nexthop entries. In Drouting, 37.4% have. The small number of nexthops in each router in Drouting, and the similarity between Deflect-1 and Drouting are interpreted as the feasibility of Drouting, since a router in Drouting architecture does not require huge memory for nexthops.

Deflect-3 fails to calculate a non-empty deflection set for 17.9% of the source-destination pairs. This is due to the stub nodes in the Telstra topology and the fact that Deflect-3 does not calculate a backtracking path.

The result seems to show that Drouting does not calculate enough nexthops to provide sufficient multipaths. However, as shown in the distribution of the number of paths (described next), Drouting provides enough multipaths. The fact that Drouting calculates the least nexthops for similar or better performance means that Drouting is more memory efficient than Deflection.

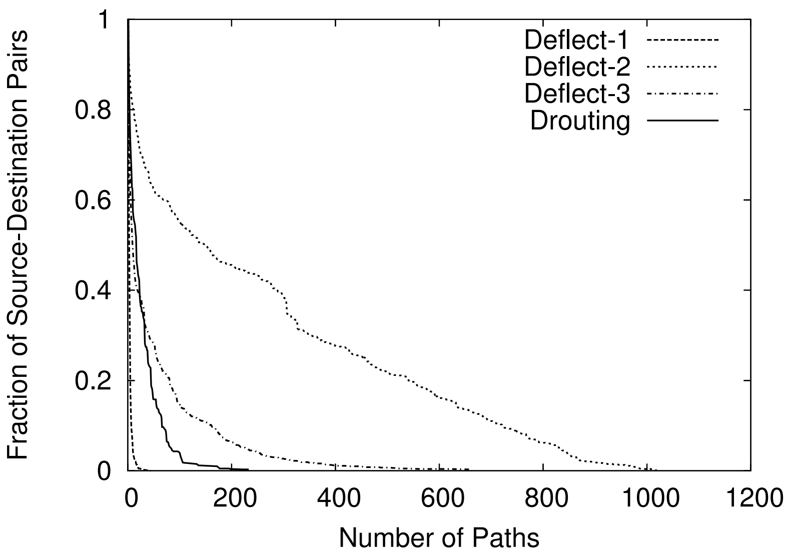

Figure 7.7: The fraction of source-destination pairs on the Telstra topology, having more than x paths.

Figure 7.7 shows the distribution of the number of paths in source-destination pairs in the network. Figure 7.8 shows the focused version of the numbers of paths for each routing method in Telstra topology, in the range below 100 paths. It shows that Deflect-1 provides only a small number of paths where almost all source destination pairs have only less than 10 paths. Although Deflect-2 has the largest number of paths, Deflect-2 is not a viable competitor because of backtracking paths as mentioned in Section 7.3. Deflect-3 and Drouting provide largely similar numbers of paths. It is surprising that even though the numbers of nexthops between Deflect-1 and Drouting is largely the same (Figure 7.6), the numbers of paths they provide are completely different. It is observed that the route

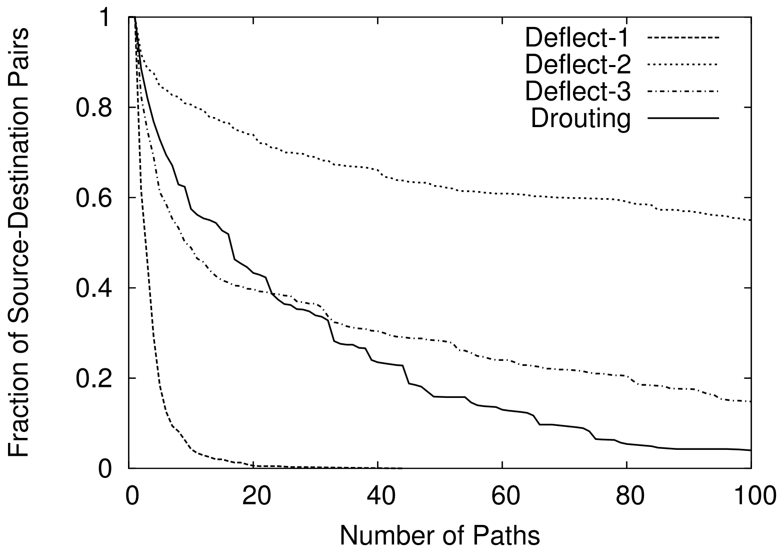

Figure 7.8: The fraction of source-destination pairs on the Telstra topology, having more than *x* paths, focused version in the range below 100 paths.

calculation method of Drouting provides a significant number of paths by using a small number of nexthops efficiently.

In Figure 7.8, Drouting provides a small number of paths (less than 20) to more fraction of source-destination pairs than Deflect-3. For example, Drouting provides more than 15 paths to 52.7% of source-destination pairs, while Deflect-3 does 41.7%. This Drouting's property is preferred, because all nodes having a similar number of alternative paths are preferred over the case where only small number of nodes have huge alternative paths. Hence, among viable competitors, Drouting was the best in this case.

Here the lengths of paths are compared for each routing methods. The shorter the length of the path becomes, the better the routing we obtain, since generally a longer path means more communication delay. The average and maximum path lengths over multipaths for each source-destination pair in Telstra are shown in Figures 7.9 and 7.10. For both average and maximum path lengths, the relations between each routing methods are largely the same. Deflect-2 tends to provide longer paths, and the maximum path length is 28 nodes. Drouting provides paths which are significantly shorter than Deflect-2 but yet slightly longer than Deflect-3. For example, the fraction of source-destination pairs that has an average path length of more than 10 nodes is 21.0% in Deflect-3, while in Drouting it is 31.7%. Deflect-1 provides shortest paths in which the length of almost all paths in all source-destination pairs are less than 10 nodes.

In summary, Drouting provides slightly longer paths in Telstra. The length of paths is completely the graph dependent, and it can be very long when Drouting is employed. This

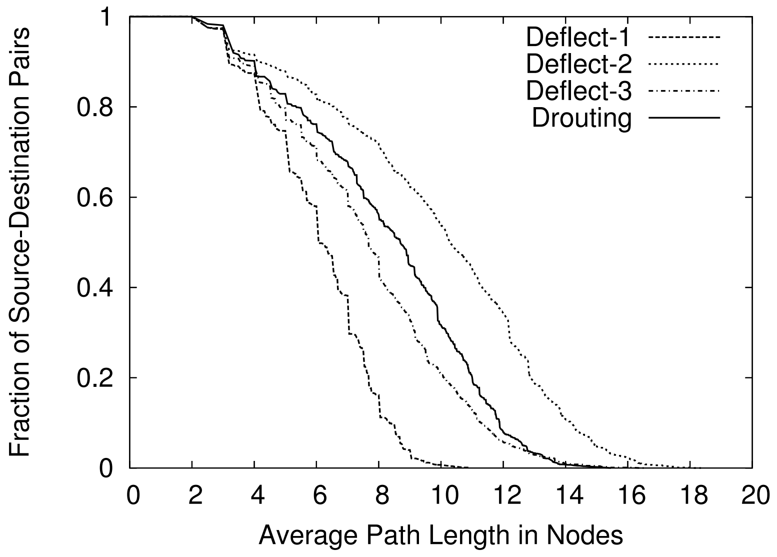

Figure 7.9: The fraction of source-destination pairs on the Telstra topology, having path length more than x in average.

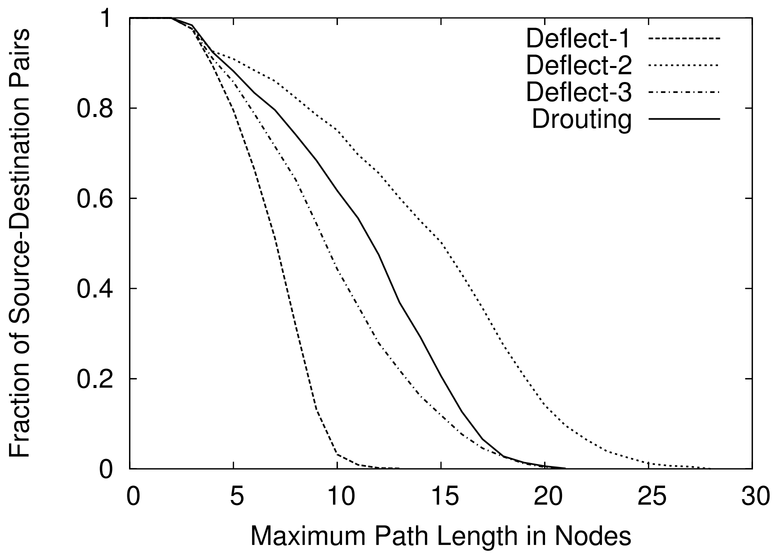

Figure 7.10: The fraction of source-destination pairs on the Telstra topology, having path length more than x in maximum.

is a trade-off for improved failure recovery capability.

7.6 Observations

Notice that although actual value increased depends on the individual network, the failure recovery probability is increased on all topologies. The Drouting improves not only the probability of recovery after ten retries, but also the probability on the first retries.

The improvement of the failure recovery property of Drouting stems from the way MA ordering calculates the multipath on the network graph. MA ordering chooses the node with the maximum connectivity to the node set that has already decided the routes to the destination, in each of its steps. The decision of routes to destination on a node with smaller connectivity is postponed, so that the node can utilize the routes on other nodes with more connectivity later when the node is decided on its routes. This way MA ordering maximizes the minimum connectivity among all the nodes to the destination (see Chapter 6). This enables a smaller number of nexthops to result in a larger number of paths to the destination using efficient utilization of combinations of the nodes with the small number of nexthops.

The number of paths in Drouting concentrate in lower numbers among all nodes, compared to Deflect-3 (Figure 7.7). Deflect-3 distributes the number of paths uniformly among nodes, namely the number of nodes with more paths and with lesser paths are both increased compared to Drouting. This negatively affects the failure recovery property, since the more nodes with lesser paths, the smaller the possibility to recover from failure at these points.

Deflection methods are based on shortest path routing, considering a smaller routing metric to be better. Hence except Deflect-2 which backtracks, Deflection chooses shorter paths and prunes longer paths. This leads to a relatively smaller number of hops in the paths. On the other hand, Drouting which uses MA ordering, does not consider routing metrics and shortness of paths, and calculates also longer, roundabout paths. The existence of longer roundabout paths appears both in average and maximum path length (Figure 7.9 and 7.10).

7.7 Summary

Drouting architecture is presented, which enables user-driven change of traffic paths. Users or end hosts can avoid failures even when the routing system fails to detect the failures. Hence Drouting contributes to improving the robustness and availability of the network.

Simulations showed improved failure recovery probability compared to the previous work called Deflection. The properties of Deflection and Drouting are studied in four aspects, namely the number of nexthops, the number of paths, the length of paths and the failure recovery probability. The results showed significant improvement in failure recovery probability while the other properties are reasonable.

There are a number of beneficial characteristics in the Drouting architecture. First, many multipath routes can be held in a Drouting network and utilized effectively. Many multipath routes are further expected to be used for the purpose of load-balancing and QoS-oriented routing. Second, calculating routes by constructing DAGs that include all links in the network can increase the probability to recover reachability. The probability increases as the network is more redundant. Third, the Drouting architecture is a simple and intuitive extension of the current routing architecture. The simplicity of the Drouting architecture preserves the most beneficial properties of the Internet such as extendability. Last, network administrators in internet service providers can enforce their routing policies by using route filters in constructing DAGs.

The future work is as follows. This chapter calculates DAGs in a centralized way. But in practice, the DAGs and multipath routes should be calculated in a distributed fashion, where each router calculates them independently and autonomously. The distributed algorithm and the protocol for the distributed computation of DAGs are left as future work. Implementation of traffic engineering and QoS routing on top of Drouting have not been addressed. Further simulations employing more complicated user traffic models have to be studied, where many users located across the network change their paths independently.

It is anticipated that the Drouting architecture can be applied also at the inter-domain routing level. Because the Drouting architecture abstracts networks in general graph structures, it can be applied to both the intra-domain level (the nodes are routers) and the inter-domain level (the nodes are Autonomous Systems (ASes)). Many issues, such as designing the new inter-domain routing protocol, are left as future work for the inter-domain routing level. In particular, the multipath routing calculation method used in this chapter, MA ordering, is not considered appropriate because it requires the synchronization of the entire topology among all nodes (in other words, synchronization of the entire Internet topology among all ASes). This is not realistic, and hence we will need a new multipath routing calculation algorithm that enables distributed computation and enforcement of AS policies.

Chapter 8

Traffic Engineering on Drouting Architecture

8.1 Overview

Congestion in the network excessively degrades communication performance. A congested network is not available to perform communication, therefore the traffic engineering capability is a required function to implement the highly available Internet.

Although many traffic engineering technologies have been studied (see Section 3.7), they have insufficient failure recovery capability, assuming only a single failure and not supporting network-wide failure recovery, thus requiring a preparation for each failures which needs precaution.

Since Drouting architecture has a split ratio for each entry in the multipath routes, optimizing the split ratios against the combination of multipath routes and the given traffic demands is possible.

The problem of routing of traffic flows on a single arbitrary path in the network graph without splitting, in the order received, while minimizing the maximum relative load imposed on network links, is NP-complete. It is also known as Unsplittable Multi-Commodity Flow (MCF) problem. However, the splittable version of this problem is known to be polynomial [17].

This indicates that the splitting of the traffic and the optimization of its ratios are important.

Assuming that the traffic demands are the union of a massive number of micro flows, and also assuming that each of the micro flows have distinct random packet tags, Drouting enables traffic splitting at the unit of the micro flow. The smaller the traffic size of a micro flow becomes, the finer granularity of splitting we obtain. With these hypotheses, the traffic engineering issue is tackled in this chapter.

A method for traffic engineering on the Drouting architecture is proposed using solving of Linear Programming (LP) problem. An LP model is borrowed from past research, and is modified to suit the Drouting architecture and to preserve the desired failure recovery capability.

With traffic engineering capability, Drouting architecture with failure recovery significantly contributes to implementing the highly available Internet. Controlling the existence of congestions by traffic engineering means some aspects of availability control over the network, thus the traffic engineering is the required function to the highly available Internet.

Section 8.2 gives a slightly modified version of the borrowed LP model. Section 8.3 presents the example optimization using the LP model. Section 8.4 tackles to preserve failure recovery capability while executing traffic engineering. Section 8.5 summarizes this chapter.

8.2 Linear Programming Model

Network optimization problem, specifically finding of the setting of near-optimal routing metric, was tackled [30] with comparison with optimal routing calculated by solving an Linear Programming (LP) problem.

Here, to present the feasibility of the traffic engineering on the Drouting architecture, the LP problem in [30] is slightly changed to fit to the Drouting architecture. The purpose of the LP problem in [30] was to find optimal traffic splitting on each node in the base graph structure of the network. On the other hand, the purpose here is to find the optimal traffic split ratios in the routing graphs for each destination calculated in the Drouting architecture. The routing graphs are the subset of the base network graph, and they represent the multipath routes in the Drouting architecture. This means that multipath routes are given in advance to the LP problem.

Network model is as follows. A directed network $G = (N, A)$ with a capacity $cap(a)$ for each arc $a \in A$ and a demand matrix D are given, where D tells the demand $D(s,t)$ between s and t for each pair $(s,t) \in N \times N$. $f_a^{(s,t)}$ tells how much of the traffic flow from s to t goes over a. $l(a)$ represents the total load on arc a, i.e., the sum of the flows going over a. Φ_a is a piecewise linear cost function that is a function of the load $l(a)$ on arc a. In Drouting architecture, multipath routes are calculated for each destination t. The multipath routes are represented by the directed set of links denoted as A_t for each $t \in N$.

Then the problem to find optimal traffic split ratio among the multipath routes are defined as follows.

Minimize:

$$\Phi = \sum_{a \in A} \Phi_a \tag{8.1}$$

subject to

$$\sum_{x:(x,y)\in A_t} f^{(s,t)}_{(x,y)} - \sum_{z:(y,z)\in A_t} f^{(s,t)}_{(y,z)} = \begin{cases} -D(s,t) & \text{if } y = s, \\ D(s,t) & \text{if } y = t, \\ 0 & \text{otherwise} \end{cases} \qquad y,s,t \in N, \tag{8.2}$$

$$l(a) = \sum_{(s,t)\in N\times N} f^{(s,t)}_a \qquad a \in A, \tag{8.3}$$

$$\Phi_a \geq l(a) \qquad a \in A, \tag{8.4}$$

$$\Phi_a \geq 3 \cdot l(a) - \frac{2}{3} \cdot cap(a) \qquad a \in A, \tag{8.5}$$

$$\Phi_a \geq 10 \cdot l(a) - \frac{16}{3} \cdot cap(a) \qquad a \in A, \tag{8.6}$$

$$\Phi_a \geq 70 \cdot l(a) - \frac{178}{3} \cdot cap(a) \qquad a \in A, \tag{8.7}$$

$$\Phi_a \geq 500 \cdot l(a) - \frac{1468}{3} \cdot cap(a) \qquad a \in A, \tag{8.8}$$

$$\Phi_a \geq 5000 \cdot l(a) - \frac{19468}{3} \cdot cap(a) \qquad a \in A, \tag{8.9}$$

$$f^{(s,t)}_a \geq 0 \qquad a \in A; s,t \in N, \tag{8.10}$$

$$fot^t_y = \sum_{s\in N,(y,z)\in A_t} f^{(s,t)}_{(y,z)} \qquad t,y \in N, \tag{8.11}$$

$$r^t_{(y,z)} = (\sum_{s\in N} f^{(s,t)}_{(y,z)})/fot^t_y \qquad (y,z) \in A; t \in N. \tag{8.12}$$

Equation 8.2 indicates that, y must produce the traffic of size $D(s,t)$ if y is the source, y must receive the traffic of size $D(s,t)$ if y is the destination, and otherwise y must relay the received traffic to other nodes in order for the traffic to eventually reach its destination. This is called flow conservation constraints, and ensures that there is no drop in forwarding traffic. Equation 8.3 calculates the load for each arcs by summing all flows traversing the arc a. From Equation 8.4 to 8.9 the model specifies the piecewise linear cost function in relation to the load on the arc. Each equation determines individual piece of the function, by specifying the slope and the y-intercept of the function for the range. The constant coefficients appeared in these equations are determined in [30].

In Equation 8.11, the total sum of the flows destined to t traversing y, fot^t_y, is calculated. Then the total sum of the flows is used in Equation 8.12 to calculate the traffic split ratio on y for nexthop z over the multipaths destined to t, which is denoted by $r^t_{(y,z)}$.

For the part from Equation 8.1 to 8.10, the only difference from LP problem in [30] is that in Equation 8.2, the links on which the traffic can be flowed is restricted by the routing graph, A_t, rather than the original A.

8.3 An Example of Traffic Engineering

A simple example of traffic engineering using LP is given in this section. The network graph is generated by BRITE with configuration file shown in Figure 8.1, and is illustrated in Figure 8.2. Note that all link bandwidth are value of 100.0. On the network graph, the routing graph for destination node 0 by Dijkstra and MARA-MC are given in Figure 8.3, for example. Dijkstra's routing metric is set to `minimum-hop` (see Section 4.5.3 in Chapter 4).

```
BriteConfig

BeginModel
  Name =  2       #Router Barabasi=2, AS Barabasi =4
  N = 4           #Number of nodes in graph
  HS = 1000       #Size of main plane (number of squares)
  LS = 100        #Size of inner planes (number of squares)
  NodePlacement = 1 #Random = 1, Heavy Tailed = 2
  m = 2           #Number of neighboring node each new node connects to.
  BWDist = 1      #Constant = 1, Uniform =2, HeavyTailed = 3, Exponential = 4
  BWMin = 100.0
  BWMax = 1000.0
EndModel

BeginOutput       #**Atleast one of these options should have value 1**
  BRITE = 1       #0 = Do not save as BRITE, 1 = save as BRITE.
  OTTER = 0       #0 = Do not visualize with Otter, 1 = Visualize
  DML = 0         #1/0=enable/disable output to SSFNet's DML format
  NS = 0          #1/0=enable/disable output to NS-2
  Javasim = 0     #1/0=enable/disable output to Javasim
EndOutput
```

Figure 8.1: BRITE configuration file for traffic engineering example.

Approximate values (rounded by 1) of traffic demands created by fortz-thorup model (see Section 4.5.5 in Chapter 4) is given in Figure 8.4. The resulting link utilizations routed by Dijkstra and MARA-MC are shown in Table 8.1 and 8.2. ECMPs in Dijkstra and multipaths in MARA-MC are equally split. Note that Dijkstra causes a link to be overloaded (link from node 1 to node 3 in Figure 8.1), but MARA-MC does not, in this case.

MINOS [94] via AMPL [3] is used to solve the LP problem described in Section 8.2. The model file is given in Figure 8.5 and 8.6, and the data file is given in Figure 8.7. The set of nodes N in Section 8.2 corresponds to `V`, A corresponds to `E`, and $\sum_{t \in N} A_t$ corresponds to `R`.

Results of the link utilizations and the value of the cost function, Φ are shown and

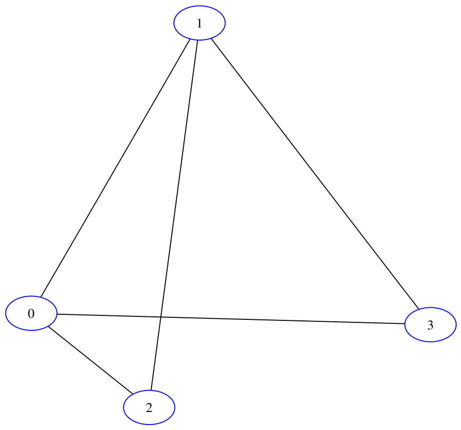

Figure 8.2: Network graph structure for traffic engineering example.

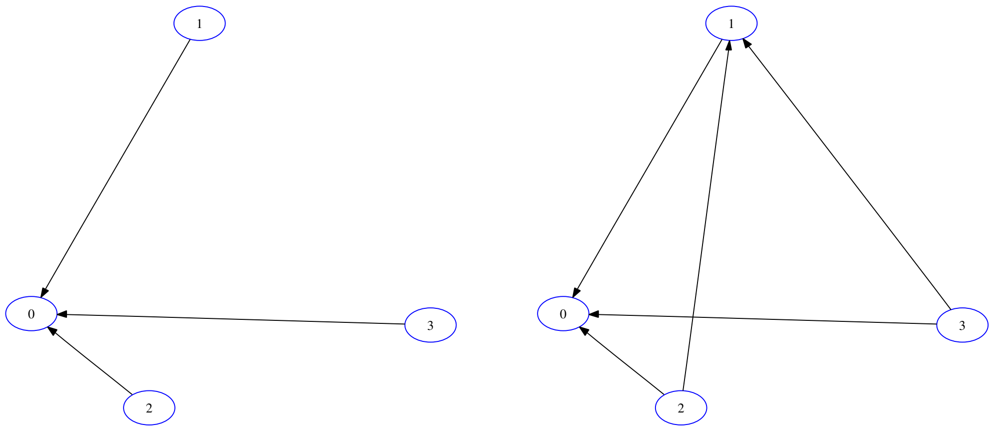

Figure 8.3: Dijkstra's and MARA-MC's routing graph for destination node 0.

```
     0    1    2    3
0    1   18    6   17
1    4   54   30  120
2    0   15   10   30
3    3   68   24   33
```

Figure 8.4: Traffic demands for traffic engineering example.

Table 8.1: Link utilizations routed by Dijkstra.

EdgeId	s	t	Load	BW	Util
0	0	1	18.139	100.000	0.181
1	1	0	3.885	100.000	0.039
2	0	2	18.121	100.000	0.181
3	2	0	15.058	100.000	0.151
4	1	2	41.679	100.000	0.417
5	2	1	29.770	100.000	0.298
6	3	0	14.941	100.000	0.149
7	0	3	32.170	100.000	0.322
8	3	1	79.770	100.000	0.798
9	1	3	134.948	100.000	1.349

Util: routing-100: max: 1.349481 min: 0.038855 med: 0.694168 avg: 0.388480 std: 0.377667

Table 8.2: Link utilizations routed by MARA-MC.

EdgeId	s	t	Load	BW	Util
0	0	1	59.483	100.000	0.595
1	1	0	93.773	100.000	0.938
2	0	2	38.960	100.000	0.390
3	2	0	22.414	100.000	0.224
4	1	2	20.840	100.000	0.208
5	2	1	22.414	100.000	0.224
6	3	0	47.355	100.000	0.474
7	0	3	99.644	100.000	0.996
8	3	1	47.355	100.000	0.474
9	1	3	67.474	100.000	0.675

Util: routing-200: max: 0.996436 min: 0.208397 med: 0.602417 avg: 0.519713 std: 0.269169

```
/* #Nodes, |V| */
param N >= 2;

/* set of vertices */
set V := {0..(N-1)};

/* set of edges */
set E within V cross V;

/* set of routing sub-graphs */
set R within V cross E;

/* there must be no loops in E */
check{(i,j) in E}: i != j;

/* there must be no loops in R */
check{(t,i,j) in R}: i != j;
/* there must be no edge from t in R */
check{(t,i,j) in R}: i != t;

param cap{E} default 0;
param D{V,V} default 0;

var F{V,V,E} >= 0, default 0.0;
var l{E} >= 0, default 0.0;
var u{E} >= 0, default 0.0;
var phi{E} >= 0, default 0.0;

var Phi >= 0;

/* ratio */
var r{V,E} >= 0, default 0.0;

var fot{V,V} >= 0, default 0.0;
var vari{V,E} >= 0, default 0.0;
var equality >= 0, default 0.0;
```

Figure 8.5: AMPL model file for traffic engineering example.

```
/* Objective */
minimize PHI: Phi;

subject to CALCPHI: sum{(x,y) in E} phi[x,y] <= Phi;

subject to FLOWSRCOUT{s in V, t in V : s != t}:
  sum{(t,s,x) in R} F[s,t,s,x] = D[s,t];

subject to FLOWDSTIN {s in V, t in V : s != t}:
  sum{(t,x,t) in R} F[s,t,x,t] = D[s,t];

subject to FLOWRESERV{t in V, y in V: t != y}:
  sum{s in V, (t,x,y) in R : s != y} F[s,t,x,y] -
  sum{s in V, (t,y,z) in R : s != y} F[s,t,y,z] = 0;

subject to GETLOAD{(x,y) in E}:
  sum{s in V, t in V} F[s,t,x,y] <= l[x,y];

subject to GETUTIL{(x,y) in E}:
  l[x,y] / cap[x,y] <= u[x,y];

subject to PHI1{(x,y) in E}: phi[x,y] >=        l[x,y];
subject to PHI2{(x,y) in E}: phi[x,y] >= 3    * l[x,y] - 2/3     * cap[x,y];
subject to PHI3{(x,y) in E}: phi[x,y] >= 10   * l[x,y] - 16/3    * cap[x,y];
subject to PHI4{(x,y) in E}: phi[x,y] >= 70   * l[x,y] - 178/3   * cap[x,y];
subject to PHI5{(x,y) in E}: phi[x,y] >= 500  * l[x,y] - 1468/3  * cap[x,y];
subject to PHI6{(x,y) in E}: phi[x,y] >= 5000 * l[x,y] - 19468/3 * cap[x,y];

subject to GETFLOWTOTAL{t in V, y in V: t != y}:
  sum{s in V, (t,y,z) in R: s != t} F[s,t,y,z] = fot[t,y];

subject to GETRATIO{(t,y,z) in R}:
  (sum{s in V: s != t} F[s,t,y,z]) / fot[t,y] = r[t,y,z];
```

Figure 8.6: AMPL model file for traffic engineering example (continue).

```
data;

param N := 4;

set E :=
 (0,1) (0,2) (0,3) (1,0) (1,2)
 (1,3) (2,0) (2,1) (3,0) (3,1)
;

param cap :=
  [0,1] 100.000000,
  [1,0] 100.000000,
  [0,2] 100.000000,
  [2,0] 100.000000,
  [1,2] 100.000000,
  [2,1] 100.000000,
  [3,0] 100.000000,
  [0,3] 100.000000,
  [3,1] 100.000000,
  [1,3] 100.000000,
;

param D   :     0     1     2     3 :=
          0  0.62 18.14  6.10 17.34
          1  3.89 54.33 29.66 120.12
          2  0.23 14.94 10.19 29.65
          3  2.92 67.75 24.05 32.81
;

set R :=
  (0,1,0) (0,2,0) (0,2,1) (0,3,0) (0,3,1)
  (1,0,1) (1,2,0) (1,2,1) (1,3,0) (1,3,1)
  (2,0,2) (2,1,0) (2,1,2) (2,3,0) (2,3,1)
  (3,0,3) (3,1,0) (3,1,3) (3,2,0) (3,2,1)
 ;
```

Figure 8.7: AMPL data file for traffic engineering example.

compared in Table 8.3. `Dijkstra` and `MARA-Equal` are the same with Table 8.1 and 8.2. `MARA-LP` is the result of solving the LP problem in Section 8.2 on the Drouting architecture. `Optimal` shows the optimal network that may not be obtained in hop-by-hop network, calculated by the LP problem in [30]. Note that the worst link utilization is 1.349 in Dijkstra, 0.996 in MARA-Equal, while 0.9 in MARA-LP and Optimal.

Table 8.3: Link utilizations comparison after solving LP.

s	t	Dijkstra	MARA-Equal	MARA-LP	Optimal
0	1	0.181	0.595	0.245033	0.245033
0	2	0.181	0.390	0.3015	0.3015
0	3	0.322	0.996	0.7711	0.7711
1	0	0.039	0.938	0.3401	0.333333
1	2	0.417	0.208	0.2966	0.303367
1	3	1.349	0.675	0.9	0.9
2	0	0.151	0.224	0.2988	0.305567
2	1	0.298	0.224	0.1494	0.1494
3	0	0.149	0.474	0.333333	0.333333
3	1	0.798	0.474	0.613867	0.613867
Φ				919.757	919.08

The split ratios among multipaths calculated by the LP solving (`var r` in Figure 8.5) are shown in Table 8.4. The split ratios in Table 8.4 do not indicate the use of significant variety of paths within multipaths. This deficiency is tackled in the next section.

8.4 Challenge to Equalize Split Ratio

The split ratios in Table 8.4 do not indicate the use of significant variety of paths within multipaths. Even if multipaths are calculated by MARA-MC, when the split ratio is 0 the path would not be used. This may affect the failure recovery property of Drouting architecture adversely. Since the failure recovery property is the primal objective for the purpose of the highly available Internet, a challenge to distribute split ratio equally is necessary.

In order to search the possibility of realizing the network optimization with more equally distributed split ratio without excessively sacrificing the link utilizations, and to distribute the split ratio equally over the unused multipaths, variances of ratios are calculated.

Let A_{ty} denote the set of links included in A_t and incident from y. Let σ_{ty}^2 denote the variance of the split ratios for the destination t on the node y, from the equal split $\overline{r_{(y,z)}^t} = 1.0/|A_{ty}|$. Followings are added to the LP model.

Table 8.4: Split ratio calculated by LP.

Node	Destination	Nexthop	Split Ratio
0	1	1	1
0	2	2	1
0	3	3	1
1	0	0	1
1	2	0	0
1	2	2	1
1	3	0	0.250749
1	3	3	0.749251
2	0	0	1
2	0	1	0
2	1	0	0
2	1	1	1
2	3	0	1
2	3	1	0
3	0	0	1
3	0	1	0
3	1	0	0.0939237
3	1	1	0.906076
3	2	0	1
3	2	1	0

Minimize:

$$\Phi + 100 \cdot \sum_{(t,y) \in N \times N} \sigma_{ty}^2 \tag{8.13}$$

subject to

$$\sigma_{ty}^2 = \sum_{(y,z) \in A_{ty}} \frac{1}{|A_{ty}|} (r_{(y,z)}^t - \overline{r_{(y,z)}^t})^2. \tag{8.14}$$

Since sum of the variances of split ratios is small, it is scaled by 100 in Equation 8.13. This may sacrifices ϕ (and thus Φ) when link utilization is low.

Corresponding addition of AMPL model file is shown in Figure 8.8. `vari[t,y]` corresponds to σ_{ty}^2, and `equality` corresponds to $100 \cdot \sum_{(t,y) \in N \times N} \sigma_{ty}^2$.

```
var vari{V,V} >= 0, default 0.0;
var equality >= 0, default 0.0;

/* Objective */
minimize PHI: Phi + equality;

subject to GETVARI{t in V, y in V: t != y}:
  (sum{(t,y,z) in R} ((r[t,y,z] - (1.0 / card{ setof{(t,y,n) in R} n}))**2))
  / (card{ setof{(t,y,n) in R} n})
  <= vari[t,y];

subject to GETEQLTY:
  (sum {t in V, y in V: t != y} vari[t,y]) * 100 <= equality;
```

Figure 8.8: Addition to AMPL model file for equalization of split ratios.

Results on link utilizations are shown in Table 8.5. `MARA-LP-EQUALIZED` is the result of the modified LP problem. It decreases the sum of the variances of split ratio by slightly sacrificing the Φ without degrading the worst link utilization 0.9. Split ratios calculated by the modified LP problem are given in Table 8.6, with comparison to previous MARA-LP. Significant improvement is observed for the split ratios among the multipaths. Further equalization of split ratios may be achieved by increasing the scale parameter (i.e., 100 in Figure 8.8) although it sacrifices Φ.

Table 8.5: Link utilizations comparison of modified LP problem.

s	t	MARA-LP	MARA-LP-EQUALIZED	Optimal
0	1	0.245033	0.333333	0.245033
0	2	0.3015	0.264767	0.3015
0	3	0.7711	0.7711	0.7711
1	0	0.3401	0.395358	0.333333
1	2	0.2966	0.333333	0.303367
1	3	0.9	0.9	0.9
2	0	0.2988	0.310432	0.305567
2	1	0.1494	0.137768	0.1494
3	0	0.333333	0.31801	0.333333
3	1	0.613867	0.62919	0.613867
Φ		919.757	948.229	919.08
$100 \cdot \sum_{(t,y)\in N\times N} \sigma_{ty}^2$		172.702	68.4735	

Table 8.6: Split ratio calculated by modified LP problem.

MARA-LP				MARA-LP-EQUALIZED			
Node	Destination	Nexthop	Split Ratio	Node	Destination	Nexthop	Split Ratio
0	1	1	1	0	1	1	1
0	2	2	1	0	2	2	1
0	3	3	1	0	3	3	1
1	0	0	1	1	0	0	1
1	2	0	0	1	2	0	0.0379087
1	2	2	1	1	2	2	0.962091
1	3	0	0.250749	1	3	0	0.268122
1	3	3	0.749251	1	3	3	0.731878
2	0	0	1	2	0	0	0.50345
2	0	1	0	2	0	1	0.49655
2	1	0	0	2	1	0	0.276355
2	1	1	1	2	1	1	0.723645
2	3	0	1	2	3	0	0.903832
2	3	1	0	2	3	1	0.0961679
3	0	0	1	3	0	0	0.573
3	0	1	0	3	0	1	0.427
3	1	0	0.0939237	3	1	0	0.163315
3	1	1	0.906076	3	1	1	0.836685
3	2	0	1	3	2	0	0.792651
3	2	1	0	3	2	1	0.207349

8.5 Summary

For the highly available Internet, congestion avoidance is mandatory. Furthermore, in order for Drouting architecture to be employed in large scale ISPs, the traffic engineering capability is imperative.

The traffic engineering on Drouting architecture is studied and is shown to be feasible in this chapter. By solving the LP problem described in this chapter and by applying the resulting split ratios among multipaths on routers, traffic engineering on Drouting architecture can be achieved.

Traffic engineering capability with failure recovery in Drouting contributes to implement the highly available Internet.

Chapter 9

Conclusion

Today, the ability to communicate on the Internet is essential to human life. Internet failures can impact our health and safety by interrupting communication such as emergency phone calls. These Internet failures must be avoided, and the Internet must be constructed as a highly available communication infrastructure.

Individual solutions for each type of failure, such as fault tolerant systems and link protection mechanisms, have been researched. However, there are still significant number of failures that cannot be avoided, such as mis-configuration of network operator, software bugs, and partial and complex hardware failures that deceive the current failure detection functions. Such failures cannot be avoided in the current Internet, and the resulting outages are sometimes long.

This dissertation targets the removal of these kinds of failures, and the improvement in the availability of the Internet. The goal was defined in Chapter 2 as the "highly available Internet", where network administrators can increase the network availability by investment in network resources. Then each component required to implement the highly available Internet was provided in this research.

A new routing simulator, called SimRouting, was described in Chapter 4 to provide a method to calculate the network availability considering the method and state of the routing system in simulations, and to evaluate a routing system. Estimation of Internet availability is important because it reveals where and how Internet availability can be increased.

BGP Flap Damping technology was applied to OSPF routing system in Chapter 5, to show an example of stabilizing a routing system. To increase the availability of the Internet, which requires stability, each network component such as a routing system, must also be stable. One deficiency of OSPF routing protocol that has been its vulnerability to oscillation problems, and a flexible method to stabilize the distributed system were shown.

A family of new multipath routing algorithms, called Maximum Alternative Routing Algorithm (MARA), was presented to provide maximum alternative routes among the entire network. The new problem of maximizing a minimum maximum flow among all nodes in the network to the destination was defined as the "all-to-one max-flow routing problem", and its connectivity version as the "all-to-one maximum connectivity routing problem". MARA was proven to be optimal for both the "all-to-one max-flow routing problem" and the "all-to-one maximum connectivity routing problem" in Chapter 6.

A new multipath routing architecture, called Drouting architecture, was shown to enable avoidance of many kinds of failures using MARA. To bind a communication session to a network path stably, it utilizes the mapping between packet tag and a communication path. The Drouting architecture was compared with related work and evaluated in terms of failure recovery property, in Chapter 7.

True Internet availability also requires adequate performance on the network path, and therefore network optimization and traffic engineering are important work. A Linear Programming (LP) problem which models the network optimization problem on Drouting architecture was given, and Chapter 8 showed that traffic engineering on Drouting is feasible by solving the LP problem. It indicates that we can gain adequate communication performance on Drouting architecture.

This dissertation provides the most basic functions required to achieve the highly available Internet, and presents the feasibility of, and a strategy toward, the highly available Internet. Approaching the highly available Internet certainly indicates an advancement in approaching the Dependable Internet. This dissertation contributes to improve the availability of the Internet by providing a new routing architecture.

Chapter 10

Future Prospects

This chapter discusses the other issues in employing the highly available Internet, and the missing parts and future work necessary for the realization of the highly available Internet.

10.1 Failure Detection

Although failure detection is left as being outside scope of this research, it is a very important function because if it takes much time to detect failure, the Expected Time to Repair (ETR) of the Internet communication reachabilities becomes larger, and hence the Internet availability degrades.

There are works which try to minimize the time taken to detect failures. Bidirectional Forwarding Detection (BFD) [41] intends to utilize a Hello packet mechanism with millisecond interval. BFD is used for current routing protocol such as OSPF and BGP, and can be used as the failure detection in the Drouting architecture as well. Another technology that is used for fast failure detection is SONET alarm [39]. It can raise an alarm within 10-20 milliseconds from loss of connectivity.

There are two desirable properties in Drouting architecture related to failure detection.

First, the failure detection can be implemented in any element along with the communication path. Because changing the packet tag changes the communication path stochastically, any of application software at end host, transport protocol at end host, gateway router, edge router, and core router can implement failure detection function and triggers switching of communication path by changing the packet tag. SCTP [96] is an example of transport protocol that may implement the failure detection. It can handle multiple communication sessions, possibly with each different packet tag. SCTP may offer the best communication path out of multiple sessions to the application software, and may change the communication path transparently to application software.

Second is that we can construct an individual failure detection function for each application service. One application may consider a communication interruption of 30 minutes interval as a failure, while another application may consider 300 milliseconds of communication delay or jitter as a failure. Another application can define complex Quality of Service (QoS) check function to decide when to change communication path. Estimating Round Trip Time (RTT) or the bandwidth (by Packet Pair [51], for example) may be used for the QoS check. The possibility of constructing an customized individual failure detection function for each application enables complex and flexible failure detection. It is free to decide a failure detection function by the application itself. They must only decide when to change the packet tag in the Drouting architecture.

10.2 Simultaneous Redundant Packet Transmission

The multiple duplicated packet transmission may be implemented on the Drouting architecture, to satisfy a certain probability of packet arrival. When some copies of a packet (for example, 5 copies) are transmitted simultaneously and redundantly with distinct packet tags, the probability of packet arrival can be increased, similar to the failure recovery simulation results presented in Chapter 7. Given the network model including the network graph structure and the failure probability of each network element, the probability of success of one of the packet arrival out of randomly chosen 5 packet tags, can be easily calculated. Given this estimated probabilities for n-copies, we can achieve a certain packet arrival probability (and hence availability) by controlling the number of copy, n.

10.3 Issues in Traffic Engineering Method

Although traffic engineering using LP presented in Chapter 8 can calculate optimal solution, generally solving the LP problem may take long time, and cannot be applied to real network in a real-time fashion. Hence, faster calculation such as by heuristic approaches must be studied.

Network optimization against exact traffic demands is known to fail when there are some errors in estimating the traffic matrix [6]. Traffic engineering that is tolerant to errors and changes in traffic matrix must be studied also for the Drouting architecture.

10.4 Relation with MPLS Technology

This section discusses how the Drouting architecture conforms to the existing MPLS technologies. The Drouting architecture has not been discussed regarding the relation with

MPLS, which is an important technology that is already deployed. This section presents three possibilities of interaction between MPLS and Drouting.

First, MPLS provides a virtual circuit that is to be used as an IP link. In other words, MPLS paths only define layer-2 topology of the network. In this case, Drouting architecture is constructed on the layer-2 topology. Traffic engineering using MPLS path setup is not executed and the traffic engineering function must be done within the Drouting architecture.

Second, MPLS computes the path for each communication session (or for each aggregated bulk communication session) considering the state of network such as link's bandwidth in order to execute MPLS traffic engineering, irrespective to layer-3 routes. The path computed by MPLS or its corresponding path computation algorithm has nothing to do with the multipath routes calculated by the Drouting architecture. When MPLS path setup fails, packet forwarding or path setup can be directed along with the layer-3 routes. In this case multipath routes calculated by the Drouting architecture are used, only as an emergency exit.

Third is the case where MPLS traffic engineering is built upon the layer-3 routing by Drouting architecture. During MPLS traffic engineering, MPLS path is setup for each communication session considering the state of network such as link's bandwidth, for example, by RSVP or OSPF. In this case the MPLS path may be constructed on the one of the path from the multipath routes provided by the Drouting architecture. This means the reduction of CPU-time consumption and computing complexity of MPLS path computation, because MPLS path can be computed from a restricted graph structure (multipath routing sub-graph calculated by the Drouting architecture) rather than the original base graph structure.

Further interaction between MPLS and Drouting, such as state distribution on the Drouting architecture that is similar to OSPF-TE, is open to research.

10.5 Network Operation and Debugging

Network operation and network debugging would become slightly complex than the current Internet, in the Drouting architecture. In Drouting architecture, it will become harder to find a network failure, since different packet tags may keep the communication from being interrupted by the failure. This means that a user or a network administrator may or may not see the failure, depending on the packet tag. It complicates locating the network failure and to determine the source of the failure.

The network administrators must use ping and/or traceroute with a specification of a packet tag, to quickly notice a problem. Hence the network administrators must execute huge number of ping and/or traceroute in order to be assured of the network state. This is the trade-off for the increased network availability.

Other network operation processes are expected to remain the same.

10.6 Migration and Transition Issues

Migration from existing network to that of Drouting architecture is not simple.

The multipath routing graphs calculate by MARAs are not consistent with the shortest path tree calculated by the existing Dijkstra algorithm, except for the MARA-SPE. Hence, coexistence of routers that perform shortest path routing and routers that perform the Drouting may provide routing loops. There are, however, some methods that is expected to enable the coexistence.

First is the dual stack approach. As shown in Chapter 7, if a dominant rule is possible such as to specify that a packet tag of value 0 indicates the shortest path routing, and other value indicates routing in Drouting architecture, then making the Drouting router also capable of shortest path routing will enable the coexistence of both routers. This approach, however, is not realistic since it strictly forces the existing router and end hosts to use the 0-valued packet tags.

Second is the overlay approach. Like migration to IPv6,the Drouting network is constructed separately from the current network, using physically different network and/or layer-3 tunnels to overlay the Drouting network on the existing network. Drouting routers must be aware of Drouting capability of neighboring routers, and be confident that there would be no routing loops when to forward packets to non-Drouting capable routers. This is similar to the relation between current IPv4 and IPv6 network, and is relatively realistic in that Drouting routers would be aware of neighboring router's Drouting capability, since there would be some sort of routing protocol for Drouting architecture that runs only among Drouting routers.

When Drouting network is enabled by either approach, there would be no problems in inter-domain routing because inter-domain routing is performed by another routing protocol, BGP, in another level. Performing the Drouting or Drouting-like routing in inter-domain routing level is another story, and is discussed in Section 10.7.

When the entire AS supports Drouting architecture, the AS may gain some benefit, even in inter-domain level without any help of neighboring ASes. Deflection [109] insists that intra-domain multipath routing that changes entire network path, such as Deflection and Drouting, can change the exit of the AS itself, hence it is likely to change the successive AS level path. This is plausible, since with the permission of the BGP routing, there may be multiple possible exit point for certain destinations outside the AS. Drouting also can be used to distribute the BGP's multiple routes, and enable end hosts and routers to switch to the other AS exit point. Hence, supporting the multipath routing mechanism such as Drouting is beneficial also in terms of utilization of AS-level alternative paths.

10.7 Inter-domain Routing

As mentioned in Section 7.7 in Chapter 7, Implementing Drouting or Drouting-like routing is difficult in inter-domain routing level, since MARA requires a complete graph of the entire network to calculate routes and to synchronize the graph with all the other nodes in the network, to guarantee loop-free multipath routes. This seems impossible, since the complete graph of the Internet seems impossible to both obtain and synchronize, even in the unit of AS.

A new breakthrough in technology is required, such as a completely new multipath routing algorithm, or a method to guarantee a loop-freeness of all possible multipath routes with only local knowledge, in order to enable multipath routing in the inter-domain routing level. It is a really challenging task and is open to research.

10.8 Improving the Routing Algorithm

One of the most short-term technical improvement that is possible would be to extend MARA to other objectives. Currently MARA only aims to maximize the minimum connectivity and the minimum maximum-flow to the destination in the network. There may be many candidates in deciding node orderings and edge orientation where any of candidates is allowed. Determining another preference on those candidates may contribute to improve the network performance. Other objectives for MARA are possible, such as maximizing reliability, minimizing delay and jitter, and the combinations of these and original objectives, namely, connectivity and bandwidth. Extending MARA is a future work.

10.9 The Distributed Routing Protocol

The discussion on Drouting architecture in this dissertation has been advanced without considering the actual distributed routing protocol that executes MARA. The Drouting architecture is assumed to utilize a link state family routing protocol such as OSPF and IS-IS, with synchronizing the network maps in a flooding procedure and each router executes the routing algorithm independently.

Link state routing protocol's correctness depends strictly on the synchronization of the link state database. A conflict in the link state database may introduce routing loops, and hence, link state routing protocols have a risky property in nature, in that if a router in the network goes out of order and advertise a faulty information, whole network may be adversely affected. In contrast, vector-based routing protocols such as distance vector's RIP and path-vector's BGP are free to reject a routing message, thus they seem to have a desired property in terms of reliability, since each router can watch and evaluate the feasibility of other routers and routes.

The improvement of routing protocols for the highly available Internet is also a future work.

Appendix A

Theory of Reliability

Network availability can be calculated as a system reliability of a *coherent system* [11]. The theory of system reliability regarding network availability is summarized below. The theory is used to calculate the availability of a real network in Chapter 4.

A.1 Systems of components

For all components in the system, a binary variable x_i is assigned to indicate the state of i-th component in the system:

$$x_i = \begin{cases} 1 & \text{if component } i \text{ is functioning,} \\ 0 & \text{if component } i \text{ is failed.} \end{cases} \tag{A.1}$$

We assume the states of the components are statistically independent.

The binary variable ϕ which indicates the state of the system is determined completely from the states of components. Hence,

$$\phi = \phi(\mathbf{x}), \tag{A.2}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ and n is the number of components in the system. The function $\phi(\mathbf{x})$ is called the *structure function*, and we denote the structure also by ϕ as in "structure ϕ".

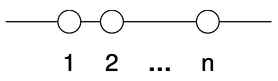

Figure A.1: A series structure.

The function of a *series* structure (Figure A.1) is given such as:

$$\phi(\mathbf{x}) = \prod_{i=1}^{n} x_i. \tag{A.3}$$

This means that all components must be functioning for the system to function.

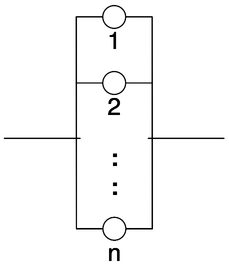

Figure A.2: A parallel structure.

$\coprod$ is defined to calculate the reliability of a *parallel* structure (Figure A.2), such that:

$$\coprod_{i=1}^{n} x_i \equiv 1 - \prod_{i=1}^{n}(1 - x_i).$$

Then the function state of a parallel structure is:

$$\phi(\mathbf{x}) = \coprod_{i=1}^{n} x_i. \tag{A.4}$$

This means that at least one of the components must be functioning for the system to function.

A.2 Coherent system

A particular set of components C is *coherent* if:

1. its structure function ϕ is increasing (i.e., non-decreasing), and

2. each component is relevant.

“Increasing” in 1 means that improving the performance of a component i always contributes (i.e., does not affect adversely) to the function state of the system $\phi(\mathbf{x})$. “Relevant” in 2 means that components whose states are irrelevant to the system state are not considered in $\mathbf{x}$. A coherent system is denoted by (C, ϕ).

Most common systems, including communication networks, are coherent.

A.3 Paths and cuts

A *path vector* is a vector $\mathbf{x}$ such that $\phi(\mathbf{x}) = 1$. Let $\mathbf{y} < \mathbf{x}$ mean $y_i \leq x_i (i = 1, \ldots, n)$, with $y_i < x_i$ for some i. Then, a *minimal path vector* is a path vector $\mathbf{x}$ such that $\mathbf{y} < \mathbf{x} \Rightarrow \phi(\mathbf{y}) = 0$. The corresponding path set $C_1(\mathbf{x}) = \{i \,|\, x_i = 1\}$, called a *minimal path set*, constitutes a minimal set of elements whose functioning insures the functioning of the system. We denote the j-th minimal path set as P_j, where $j = 1, \ldots, p$, and p is the number of minimal path sets of ϕ. Since the system functions if all components in the minimal path set function, the minimal path set can be thought as a series structure. A binary function

$$p_j(\mathbf{x}) = \prod_{i \in P_j} x_i, \tag{A.5}$$

tests if all components in the path set function. Since the system functions if and only if at least one of the minimal path structures is functioning, the state of the system can be written as

$$\phi(\mathbf{x}) \equiv \coprod_{j=1}^{p} p_j(\mathbf{x}) \equiv \coprod_{j=1}^{p} \prod_{i \in P_j} x_i. \tag{A.6}$$

This is a parallel arrangement of the minimal path series structures.

A *cut vector* is a vector $\mathbf{x}$ such that $\phi(\mathbf{x}) = 0$. A *minimal cut vector* is a cut vector $\mathbf{x}$ such that for all $\mathbf{y} > \mathbf{x} \Rightarrow \phi(\mathbf{y}) = 1$. The corresponding cut set $C_0(\mathbf{x}) = \{i \,|\, x_i = 0\}$, called a *minimal cut set*, is a minimal set of elements whose failure causes the system to fail. We denote the j-th minimal cut set as K_j, where $j = 1, \ldots, k$, and k is the number of minimal cut sets of ϕ. Since the system fails if all components in the minimal path set fail, the minimal cut set can be thought as a parallel structure. A binary function

$$k_j(\mathbf{x}) = \coprod_{i \in K_j} x_i, \tag{A.7}$$

takes the value 0 if all components in the cut set fail, and 1 otherwise. Since the system functions if and only if all the minimal cut sets function, the state of the system can be written as

$$\phi(\mathbf{x}) \equiv \prod_{j=1}^{k} k_j(\mathbf{x}). \equiv \prod_{j=1}^{k} \coprod_{i \in K_j} x_i. \tag{A.8}$$

This is a series arrangement of the minimal cut parallel structures.

A.4 System reliability

Now we extend the discussion from the state to the probability that a component functions. Suppose that the state X_i of the i-th component is random with

$$P[X_i = 1] = p_i = EX_i \qquad \text{for } i = 1, \ldots, n, \tag{A.9}$$

where EX denotes the expected value of the random variable X. p_i is the probability that i functions, i.e., the *reliability* of i. The reliability of the system is given by:

$$P[\phi(X) = 1] = h = E\phi(X). \tag{A.10}$$

The system reliability h is a function of component reliabilities:

$$h = h(\mathbf{p}). \tag{A.11}$$

$h(\mathbf{p})$ is referred to as *reliability function* of structure ϕ. The series structure $\phi(\mathbf{x}) = \prod_{i=1}^{n} x_i$ has reliability function

$$h(\mathbf{p}) = \prod_{i=1}^{n} p_i, \tag{A.12}$$

and the parallel structure $\phi(\mathbf{x}) = \coprod_{i=1}^{n} x_i$ has reliability function

$$h(\mathbf{p}) = \coprod_{i=1}^{n} p_i \equiv 1 - \prod_{i=1}^{n}(1 - p_i). \tag{A.13}$$

The system reliability can be computed simply by expanding the system function state represented by minimal path or cut sets into the x_i's multinomial expressions, using the idempotency of x_i (that is, $x_i^2 = x_i$), and taking the expectation; that is:

$$h(\mathbf{p}) = E\coprod_{j=1}^{p}\prod_{i\in P_j} X_i \tag{A.14}$$

and

$$h(\mathbf{p}) = E\prod_{j=1}^{k}\coprod_{i\in K_j} X_i. \tag{A.15}$$

An example of reliability computation for a complex system where minimal path sets share some common links follows.

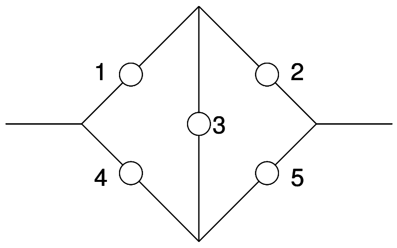

Figure A.3: An example of reliability computation by minimal path sets.

In the system of Figure A.3, the minimal path sets are:

$$P_1 = \{1,2\}, P_2 = \{1,3,5\}, P_3 = \{4,3,2\}, P4 = \{4,5\}.$$

Structure function is:

$$\begin{aligned}
\phi(\mathbf{x}) &= \coprod_{j=1}^{p} \prod_{i \in P_j} x_i \\
&= 1-(1-x_1x_2)(1-x_1x_3x_5)(1-x_4x_3x_2)(1-x_4x_5) \\
&= x_1x_2 + x_1x_3x_5 + x_4x_5 + x_2x_3x_4 \\
&\quad -x_1x_2^2x_3x_4 - x_1^2x_2x_3x_5 - x_1x_2x_4x_5 - x_1x_3x_4x_5^2 - x_2x_3x_4^2x_5 \\
&\quad -x_1x_2x_3^2x_4x_5 + x_1x_2x_3^2x_4^2x_5^2 \\
&\quad +x_1x_2^2x_3x_4^2x_5 + x_1^2x_2x_3x_4x_5^2 \\
&\quad +x_1^2x_2^2x_3^2x_4x_5 - x_1^2x_2^2x_3^2x_4^2x_5^2 \\
&= x_1x_2 + x_1x_3x_5 + x_4x_5 + x_2x_3x_4 \\
&\quad -x_1x_2x_3x_4 - x_1x_2x_3x_5 - x_1x_2x_4x_5 - x_1x_3x_4x_5 - x_2x_3x_4x_5 \\
&\quad -x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5 \\
&\quad +x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5 \\
&\quad +x_1x_2x_3x_4x_5 - x_1x_2x_3x_4x_5 \qquad \text{(idempotency)} \\
&= x_1x_2 + x_1x_3x_5 + x_4x_5 + x_2x_3x_4 \\
&\quad -x_1x_2x_3x_4 - x_1x_2x_3x_5 - x_1x_2x_4x_5 - x_1x_3x_4x_5 - x_2x_3x_4x_5 \\
&\quad +2x_1x_2x_3x_4x_5. \qquad \text{(A.16)}
\end{aligned}$$

A vector term such as $x_1x_2x_4x_5$ is called a *Boolean product.* A Boolean product representing a minimal path set or a minimal cut set, such as x_1x_2, is called *minproduct.*

The reliability can be derived by applying the probability in Equation A.16:

$$\begin{aligned}
h(\mathbf{p}) &= E\phi(\mathbf{x}) \\
&= p_1p_2 + p_1p_3p_5 + p_4p_5 + p_2p_3p_4 \\
&\quad -p_1p_2p_3p_4 - p_1p_2p_3p_5 - p_1p_2p_4p_5 - p_1p_3p_4p_5 - p_2p_3p_4p_5 \\
&\quad +2p_1p_2p_3p_4p_5. \qquad \text{(A.17)}
\end{aligned}$$

A.5 Sum of disjoint products approach

Computing the system reliability by Equation A.14 involves computation of $O(2^p)$ terms, which is computationally intractable. To address the problem, numerical researches have been proposed including the *Sum of Disjoint Products* (SDP) approach [80]. Some of the

SDP methods utilize Multiple Variable Inversion (MVI) techniques, where a *cube* represents a state of components that indicates aggregation of multiple events. A cube $A = a_1a_2\dots a_n$, where

$$a_i = \begin{cases} 1 & \text{if } x_i \text{ is true,} \\ x & \text{if } x_i \text{ does not matter,} \\ 0_u & \text{if } a_i \text{ is involved in grouped zeros labeled by } u. \\ & \text{This indicates a state that at least one component in the group } u \text{ fail.} \end{cases} \tag{A.18}$$

0_u enables the inversion on multiple variables (MVI). For example, the cube $A = a_1a_2a_3a_4a_5a_6a_7 = 0_110_210_21x$ indicates a probability of some event E_i of which the state probability is represented by $P(E) = (1-p_1)p_2p_4p_6(1-p_3p_5)$. In this example, $a_1 = 0_1$ and there is no 0_u where $u = 1$, the probability regarding x_1 is $(1-p_1)$. For x_2, x_4, and x_6, $a_i = 1$, hence the probability is $p_2p_4p_6$. For x_3 and x_5, $a_i = 0_2$, thus the probability is $(1-p_3p_5)$ (i.e., grouped inversion). x_7 does not matter because $a_7 = x$. The cube A is 0 if the resulting probability is 0.

Let E_j be the event that all components in a minimal path set P_j function (i.e., the path P_j operates). Let $\overline{E_j}$ denote the complement of event E_j, and $P(E_j)$ be the probability function of event E_j. Then reliability function is

$$h(\mathbf{p}) = P(E_1) + P(\overline{E_1}E_2) + \cdots + P(\overline{E_1E_2\dots E_{p-1}}E_p). \tag{A.19}$$

In this equation the reliability is calculated by adding the probability of P_i operates with all $P_1, P_2, \dots, P_{i-1}$ fail. Notice that these events are mutually disjoint.

Equation A.19 can be translated as

$$h(\mathbf{p}) = P(E_1) + P(E_2 \cdot \overline{E_1}) + P((E_3 \cdot \overline{E_1}) \cdot \overline{E_2}) + \cdots + P((\dots((E_p \cdot \overline{E_1}) \cdot \overline{E_2}) \cdot \dots) \cdot \overline{E_{p-1}}). \tag{A.20}$$

Hence calculation of cubes $A \cdot \overline{B}$ is important. Note that B can have only $1, x$ (i.e., cannot have 0_u) because B is an original minproduct. Hence $A = \{a_i \,|\, a_i \in \{1, x, 0_u\}\}$, $B = \{b_i \,|\, b_i \in \{1, x\}\}$.

Improved version of Veeraraghavan-Trivedi (VT), called I_VT [101], develops a new operator $\natural$ (pronounced 'little') for cubes to incrementally compute the disjoint state after some events. $P(E_1 \natural E_2)$ computes $P(E_1 \cdot \overline{E_2})$.

The operator $\natural$ requires classifying the relation of two operand cubes, A, B, into four classes: subset, disjoint, x1, and split-recursive.

1. subset

   A is a subset of B if for all i either $a_i = b_i$ or (if $a_i \neq b_i$) $b_i = x$. In this case, $A \subseteq B$, hence $A \cdot \overline{B} = 0$.

2. disjoint

   A and B are disjoint if there exist u such that for all $a_i = 0_u$, the $b_i = 1$. In this case, $A \cdot \overline{B} = A$.

3. x1

   A and B are in x1 relation if there is an index i such that $a_i = x$ and $b_i = 1$, and for all the other index j, if there exist j such that $a_j = 0_u$ and $b_j = x$. This differs from subset relation only in that there is a coordinate pair $(a_i = x, b_i = 1)$. In this case $A \cdot \overline{B} = C$. C can be obtained by slightly limiting the range of case that A represents, by substituting $a_i = x$ in A with $a_i = 0_u$ where $(a_i = x, b_i = 1)$.

   $C = c_1 c_2 \dots c_n$, where

   $$c_i = \begin{cases} 0_{(\gamma+1)} & \text{if } a_i = x,\, b_i = 1, \\ a_i & \text{otherwise.} \end{cases} \tag{A.21}$$

   γ indicates the largest label of grouped zeros in the corresponding cube. $(\gamma + 1)$ means the assignment of a new label to the new grouped zeros.

4. split-recursive

   A and B are in split-recursive relation if there is a coordinate pair $(a_i = 0_u, b_i = 1)$ at index i, and for every 0_u in cube A there is a coordinate pair $(a_j = 0_u, b_j = x)$ at index $j \neq i$. Then the 0_u part in the A is split to 1 and x, so that $A = A_1 + A_x$. A_1 is obtained from A by setting all a_i that have $(a_i = 0_u, b_i = 1)$ pairs to 1. A_x is obtained from A by setting all a_i that have $(a_i = 0_u, b_i = x)$ pairs to x. The corresponding Boolean expression is $A = \overline{x_1 \overline{x_2}} = x_1 \overline{x_2} + \overline{x_1}$. Since A_x and B constitute a disjoint relation, $A \cdot \overline{B} = (A_1 \cdot \overline{B}) + (A_x \cdot \overline{B}) = (A_1 \cdot \overline{B}) + A_x$. All 0_vs other than 0_u are handled recursively by $(A_1 \cdot \overline{B})$.

The operator $\natural$ is defined by:

$$A \natural B = \begin{cases} 0 & \text{if subset relation holds,} \\ A & \text{if disjoint relation holds,} \\ C & \text{if x1 relation holds,} \\ (A_1 \cdot \overline{B}) + A_x & \text{if split-recursive relation holds.} \end{cases} \tag{A.22}$$

$(A_1 \cdot \overline{B})$ in split-recursive relation is expanded by the operator $\natural$ further, recursively.

Appendix B

Simulations and analysis on other topologies

The result of failure recovery simulations and path analysis on the topology of Sprint, Ebone, Tiscali, Exodus, Abovenet, BA-100-invcap and BA-100-minhop are given here. The simulation method and the meaning of the path analysis are described in Chapter 7. For each topology, the illustration of the topology, failure recovery simulation results for a single failure and 10 failures, the distribution of the number of nexthops and paths, and the distribution of the average and the maximum path length among nodes are given.

The topology of Sprint is a large network graph. The comparisons of paths in such a large network is hard, because calculating the number of path in Drouting involves the enumeration of all paths (see Section 7.5.2), which does not terminate in polynomial time. Hence, the comparison of paths on Sprint is omitted. Here only the result of the failure recovery simulation is shown for Sprint.

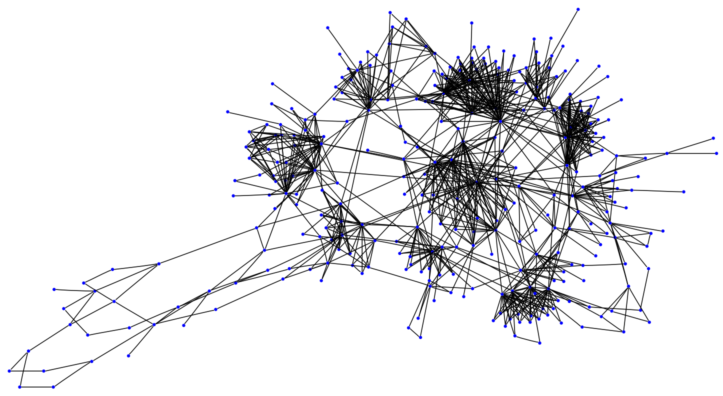

Figure B.1: The Sprint's network topology.

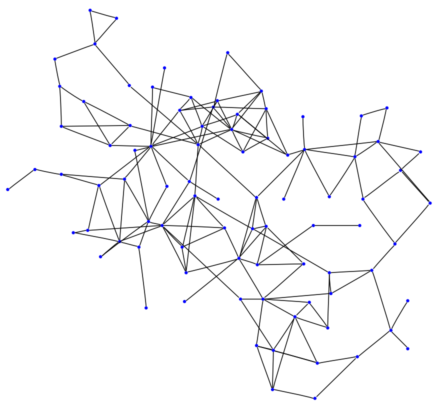

Figure B.2: The Ebone's network topology.

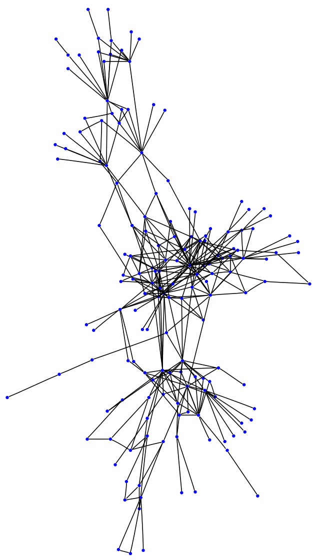

Figure B.3: The Tiscali's network topology.

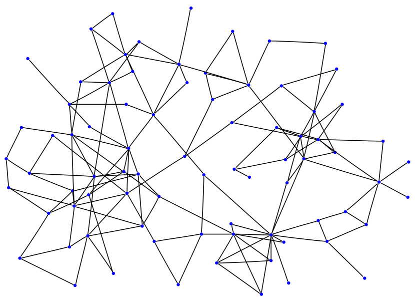

Figure B.4: The Exodus's network topology.

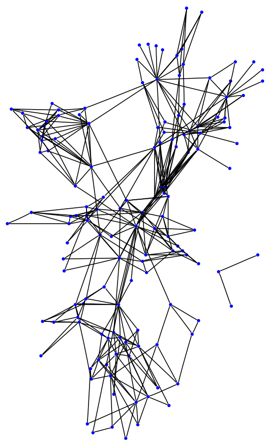

Figure B.5: The Abovenet's network topology.

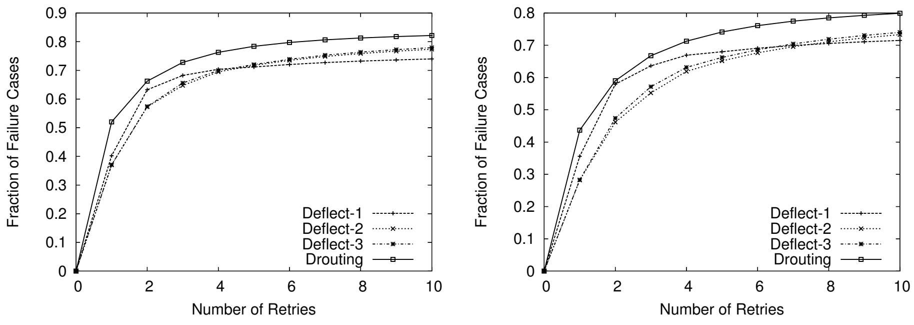

Figure B.6: The fraction of failures, successfully recovered by different systems on the Sprint topology for 1 failure (left) and 10 failures (right).

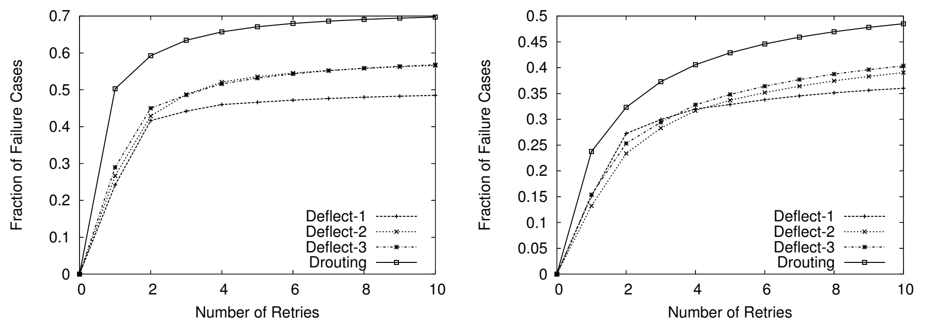

Figure B.7: The fraction of failures, successfully recovered by different systems on the Ebone topology for 1 failure (left) and 10 failures (right).

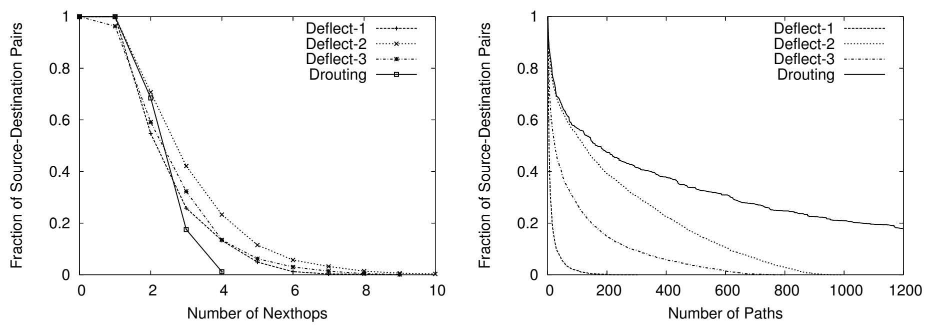

Figure B.8: The fraction of source-destination pairs on the Ebone topology, having more than *x* nexthops and paths.

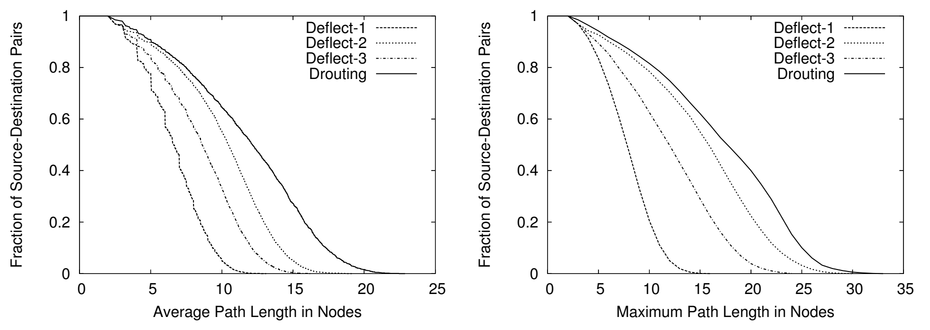

Figure B.9: The fraction of source-destination pairs on the Ebone topology, having path length more than *x* in average and in maximum.

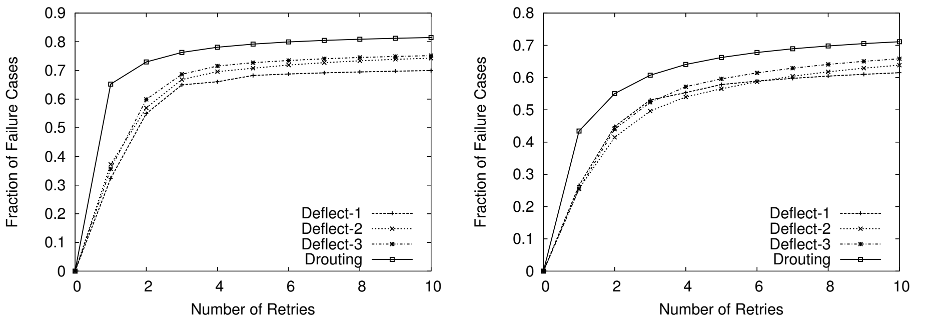

Figure B.10: The fraction of failures, successfully recovered by different systems on the Tiscali topology for 1 failure (left) and 10 failures (right).

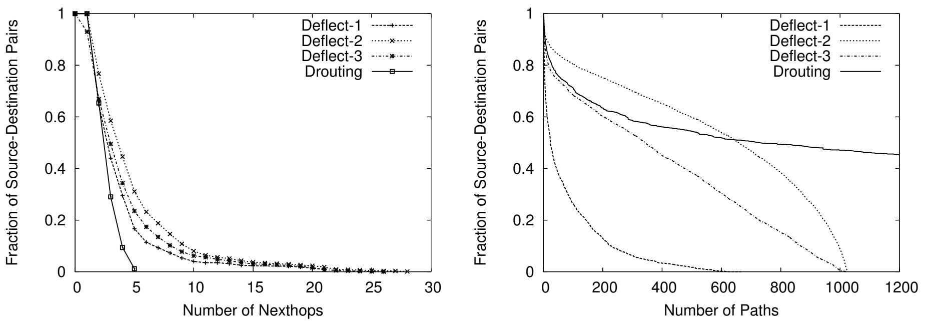

Figure B.11: The fraction of source-destination pairs on the Tiscali topology, having more than x nexthops and paths.

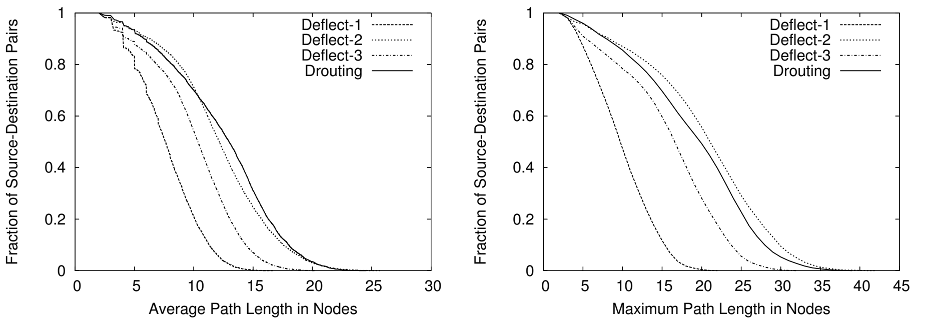

Figure B.12: The fraction of source-destination pairs on the Tiscali topology, having path length more than x in average and in maximum.

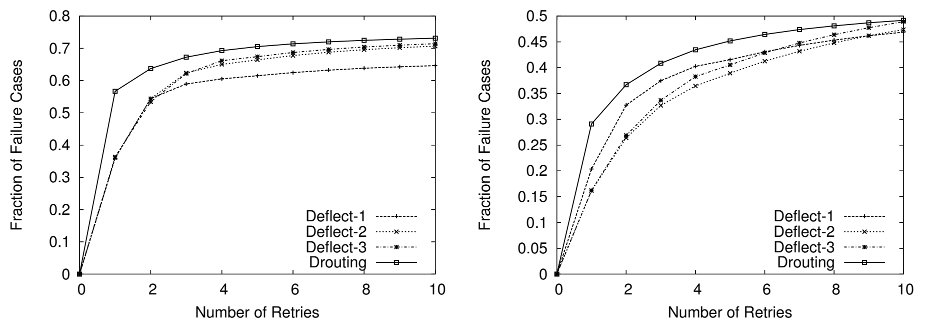

Figure B.13: The fraction of failures, successfully recovered by different systems on the Exodus topology for 1 failure (left) and 10 failures (right).

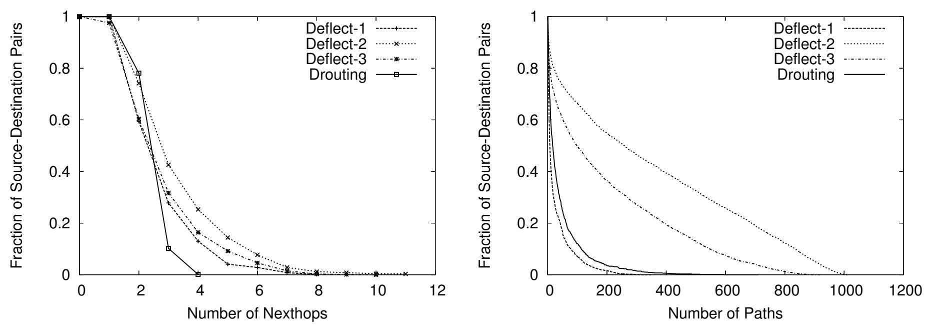

Figure B.14: The fraction of source-destination pairs on the Exodus topology, having more than *x* nexthops and paths.

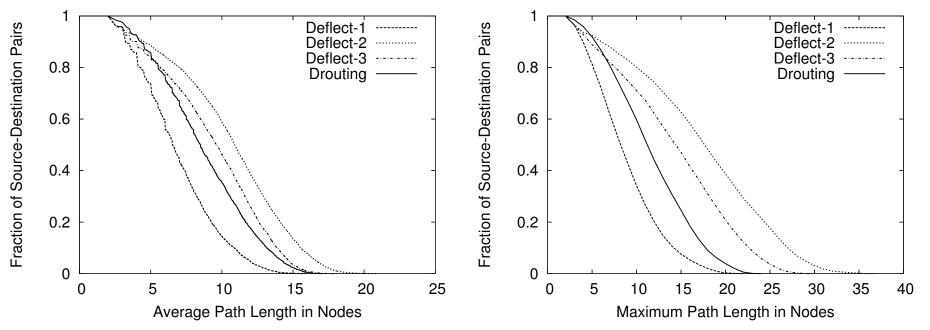

Figure B.15: The fraction of source-destination pairs on the Exodus topology, having path length more than *x* in average and in maximum.

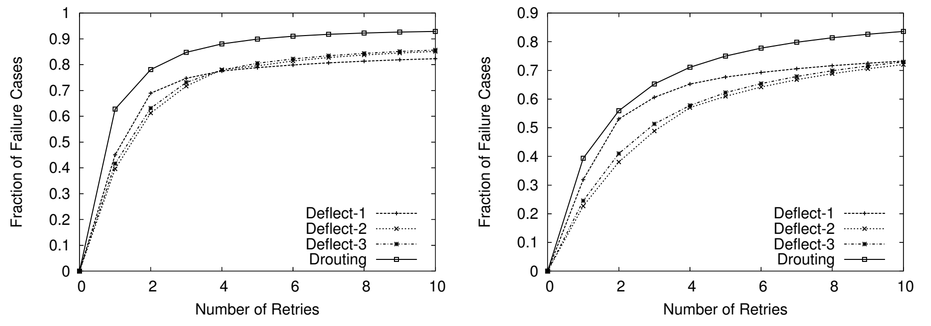

Figure B.16: The fraction of failures, successfully recovered by different systems on the Abovenet topology for 1 failure (left) and 10 failures (right).

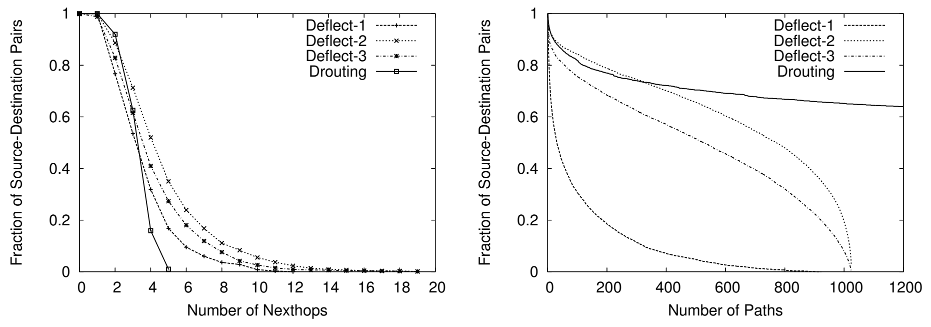

Figure B.17: The fraction of source-destination pairs on the Abovenet topology, having more than *x* nexthops and paths.

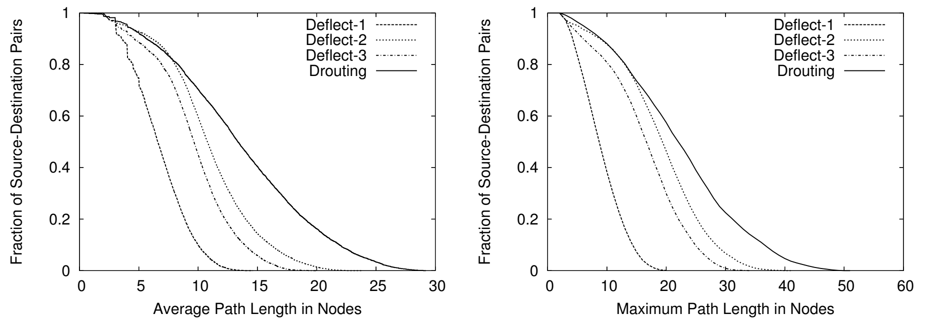

Figure B.18: The fraction of source-destination pairs on the Abovenet topology, having path length more than *x* in average and in maximum.

Bibliography

[1] *Intermediate system to Intermediate system routeing information exchange protocol for use in conjunction with the Protocol for providing the Connectionless-mode Network Service (ISO 8473).* ISO/IEC 10589:2001.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[3] ampl.com. AMPL Modeling Language for Mathematical Programming. <`http://www.ampl.com/`>.

[4] Analytical Graphics, Inc. Analytical Graphics, Inc. (AGI), analysis software for land, sea, air, and space. <`http://www.agi.com/`>.

[5] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP '01: Proceedings of the 18th ACM Symposium on Operating Systems Principles*, pp. 131–145, New York, NY, USA, 2001. ACM.

[6] D. Applegate and E. Cohen. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 313–324, New York, NY, USA, 2003. ACM Press.

[7] A. Atlas and A. Zinin. Basic specification for IP fast-reroute: loop-free alternates. Technical report, Mar. 2007. draft-ietf-rtgwg-ipfrr-spec-base-06.txt.

[8] A. Avižienis, J.-C. Laprie, and B. Randell. Fundamental Concepts of Dependability. Research Report 1145, LAAS-CNRS, 2001.

[9] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. RFC 2702, Sept. 1999.

[10] D. O. Awduche. MPLS and traffic engineering in IP networks. *Communications Magazine, IEEE*, 37(12):42–47, Dec. 1999.

[11] R. Barlow and F. Proschan. *Statistical theory of reliability and life testing: probability models.* Holt, Rinehart and Winston, New York, 1975.

[12] A. Basu, A. Lin, and S. Ramanathan. Routing Using Potentials: A Dynamic Traffic-Aware Routing Algorithm. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 37–48, New York, NY, USA, 2003. ACM Press.

[13] I. V. Beijnum. *BGP.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2002.

[14] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine. A Framework for Integrated Services Operation over Diffserv Networks. RFC 2998, Nov. 2000.

[15] K. Cho. *The Design and Implementation of the ALTQ Traffic Management System.* PhD thesis, Keio University, Jan. 2001.

[16] Cisco Systems. Advanced topics in MPLS-TE deployment. White paper, 2001.

[17] R. Cohen and G. Nakibly. On the Computational Complexity and Effectiveness of N-hub Shortest-Path Routing. In *INFOCOM*, 2004.

[18] R. Coltun, D. Ferguson, and J. Moy. OSPF for IPv6. RFC 2740, Dec. 1999.

[19] D. E. Comer. Principles, Protocols, and Architectures. In *Internetworking with TCP/IP*, volume 1. Prentice Hall PTR, Upper Saddle River, NJ, USA, fourth edition, 2000.

[20] S. Cowley. Level 3, Cogent resolve peering spat, renew deal. *IDG News Service*, Oct 28 2005. <`http://www.networkworld.com/edge/news/2005/102805-cogent-level3.html`>.

[21] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet. RFC 2386, Aug. 1998.

[22] G. B. Dantzig and D. R. Fulkerson. *On the max flow min cut theorem of networks.* The RAND Corporation, Santa Monica, California, 1955. Paper P-826.

[23] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[24] E. W. Dijkstra and C. S. Scholten. Termination Detection for Diffusing Computations. *Inf. Process. Lett.*, 11(1):1–4, 1980.

[25] E. A. Dinits. Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation. *Soviet Mathematics-Doklady*, 11(5):1277–1280, 1970.

[26] J. Duffy. Cisco routers caused major outage in Japan: report. *Network World*, May 16 2007. <`http://www.networkworld.com/news/2007/051607-cisco-routers-major-outage-japan.html`>.

[27] J. Ellson and E. Gansner. Graphviz. <`http://www.graphviz.org/`>.

[28] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: multipath adaptive traffic engineering. *Comput. Networks*, 40(6):695–709, 2002.

[29] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 251–262, New York, NY, USA, 1999. ACM Press.

[30] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *IEEE INFOCOM*, volume 2, pp. 519–528, Mar 2000.

[31] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS Weights in a Changing World. *IEEE J. Selected Areas in Communications*, 20(4):756–767, 2002.

[32] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987.

[33] A. Fumagalli and L. Valcarenghi. IP Restoration vs. WDM Protection: Is There an Optimal Choice ? *IEEE Network*, 14(6), Dec. 2000.

[34] R. Govindan and A. Reddy. An Analysis of Internet Inter-Domain Topology and Route Stability. In *INFOCOM '97: Proceedings of the INFOCOM '97. 16th Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, p. 850, Washington, DC, USA, 1997. IEEE Computer Society.

[35] T. G. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Netw.*, 10(2):232–243, 2002.

[36] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411, Dec. 2002.

[37] C. Huitema. *Routing in the Internet.* Prentice Hall PTR, Upper Saddle River, NJ, USA, second edition, 2000.

[38] G. Huston. CIDR Report. <`http://www.cidr-report.org/`>.

[39] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot. Feasibility of IP restoration in a tier 1 backbone. *IEEE Network*, 18(2):13–19, 2004.

[40] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an IP backbone. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pp. 237–242, New York, NY, USA, 2002. ACM Press.

[41] IETF bfd working group. Bidirectional Forwarding Detection (bfd). <`http://www.ietf.org/html.charters/bfd-charter.html`>.

[42] IETF manet working group. Mobile Ad-hoc Networks (manet). <`http://www.ietf.org/html.charters/manet-charter.html`>.

[43] Internet Systems Consortium, Inc. ISC Internet Domain Survey. <`http://www.isc.org/ops/ds/`>.

[44] Internet World Stats. Internet Growth Statistics - Global Village Online. <`http://www.internetworldstats.com/emarketing.htm`>.

[45] M. Jain and C. Dovrolis. Ten fallacies and pitfalls on end-to-end available bandwidth estimation. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 272–277, New York, NY, USA, 2004. ACM Press.

[46] P. Jalote. *Fault Tolerance in Distributed Systems.* Prentice Hall, Englewood Cliffs, NJ, USA, 1994.

[47] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: responsive yet stable traffic engineering. *SIGCOMM Comput. Commun. Rev.*, 35(4):253–264, 2005.

[48] T. Kenyon. *High Performance Data Network Design.* Digital Press, Elsevier Science Ltd., The Boulevard, England, 2001.

[49] C. Labovitz, A. Ahuja, and F. Jahanian. Experimental Study of Internet Stability and Wide-Area Backbone Failures. Technical Report CSE-TR-382-98, University of Michigan, 1998.

[50] C. Labovitz, A. Ahuja, and F. Jahanian. Experimental Study of Internet Stability and Backbone Failures. In *FTCS '99: Proceedings of the Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*, pp. 278+, Washington, DC, USA, 1999. IEEE Computer Society.

[51] K. Lai and M. Baker. Measuring link bandwidths using a deterministic model of packet delay. In *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 283–294, New York, NY, USA, 2000. ACM Press.

[52] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah. Proactive vs Reactive Approaches to Failure Resilient Routing. In *INFOCOM*, 2004.

[53] L. Li, D. Alderson, W. Willinger, and J. Doyle. A first-principles approach to understanding the internet's router-level topology. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 3–14, New York, NY, USA, 2004. ACM Press.

[54] G. Malkin. RIP Version 2. RFC 2453, Nov. 1998.

[55] E. Mannie and D. Papadimitriou. Recovery (Protection and Restoration) Terminology for Generalized Multi-Protocol Label Switching (GMPLS). RFC 4427, Mar. 2006.

[56] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot. Characterization of failures in an IP backbone network. In *INFOCOM*, 2004.

[57] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: an approach to universal topology generation. *MASCOTS*, 00:0346, Aug. 2001.

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

[59] J. T. Moy. OSPF version 2. RFC 2328, Apr. 1998.

[60] B. Murray. GTNetS - Home. <`http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/index.html`>.

[61] H. Nagamochi and T. Ibaraki. Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM J. Discret. Math.*, 5(1):54–66, 1992.

[62] H. Nagamochi and T. Ibaraki. Graph connectivity and its augmentation: applications of MA orderings. *Discrete Appl. Math.*, 123(1-3):447–472, 2002.

[63] A. Nakao, L. Peterson, and A. Bavier. A routing underlay for overlay networks. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 11–18, New York, NY, USA, 2003. ACM.

[64] T. Nolle. What we should learn from the AT&T outage. *Network World*, 15:73, May 4 1998.

[65] ns-2 project. The Network Simulator - ns-2. <http://www.isi.edu/nsnam/ns/>.

[66] NTT-EAST. About the circumstances where FLET'S service and Hikari phones cannot be used (final report) (Japanese). <http://www.ntt-east.co.jp/release/0705/070516b.html>.

[67] Y. Ohara. Zebra OSPFv3. <http://www.sfc.wide.ad.jp/˜yasu/research/ospf-v3.html>.

[68] Y. Ohara, H. Imaizumi, A. Kato, O. Nakamura, and J. Murai. A Load Balancing Routing Algorithm Tolerant to Wide Range of Traffic Demands (Japanese). *IPSJ*, 48(4):1627–1640, 2007.

[69] OMNeT++ Community Site. OMNeT++ Community Site. <http://www.omnetpp.org/>.

[70] OneSAF. OneSAF Public Site. <http://www.onesaf.net/>.

[71] OPNET Technologies, Inc. OPNET Technologies, Inc. <http://www.opnet.com/>.

[72] P. Pan, G. Swallow, and A. Atlas. Fast Reroute Extensions to RSVP-TE for LSP Tunnels. RFC 4090, May 2005.

[73] R. Perlman. *Interconnections: Bridges and Routers.* Addison Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1992.

[74] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the Internet. *SIGCOMM Comput. Commun. Rev.*, 33(1):59–64, 2003.

[75] T. C. Piliouras. *Network Design: Management and Technical Perspectives.* Auerbach Publications, Boston, MA, USA, 2004.

[76] J. B. Postel. User Datagram Protocol. RFC 0768, Aug. 1980.

[77] J. B. Postel. Internet Protocol. RFC 0791, Sept. 1981.

[78] J. B. Postel. Transmission Control Protocol. RFC 0793, Sept. 1981.

[79] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. *IEEE/ACM Trans. Netw.*, 14(4):725–738, 2006.

[80] S. Rai, M. Veeraraghavan, and K. S. Trivedi. A Survey of Efficient Reliability Computation Using Disjoint Products Approach. *Networks*, 25(3):147–163, 1995.

[81] J. Rajahalme, A. Conta, B. Carpenter, and S. Deering. IPv6 flow label specification. RFC 3697, Mar. 2004.

[82] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (BGP-4). RFC 4271, Jan. 2006.

[83] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031, Jan. 2001.

[84] E. C. Rosen. Vulnerabilities of network control protocols: an example. *SIGCOMM Comput. Commun. Rev.*, 11(3):10–16, 1981.

[85] L. Sahasrabuddhe, S. Ramamurthy, and B. Mukherjee. Fault management in IP-over-WDM networks: WDM protection versus IP restoration. *IEEE Journal on Selected Areas in Communications*, 20(1), Jan. 2002.

[86] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 289–299, New York, NY, USA, 1999. ACM Press.

[87] Scalable Network Technologies. Scalable Network Technologies: Creators of QualNet Network Simulator Software. <`http://www.scalable-networks.com/`>.

[88] A. Schmid and C. Steigner. Avoiding Counting To Infinity In Distance Vector Routing. *Telecommunication Systems*, 19(3–4):497–514, 2002.

[89] R. Sedgewick. *Algorithms in C*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[90] J. Sellens. *System and Network Administration for Higher Reliability.* The USENIX Association, Berkeley, CA, USA, 2001.

[91] J. L. Sobrinho. Network routing with path vector protocols: theory and applications. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 49–60, New York, NY, USA, 2003. ACM Press.

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

[93] A. Sridharan, R. Guérin, and C. Diot. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. *IEEE/ACM Trans. Netw.*, 13(2):234–247, 2005.

[94] Stanford Business Software, Inc. MINOS 5.5. <`http://www.sbsi-sol-optimize.com/asp/sol_product_minos.htm`>.

[95] J. A. Stankovic. *Reliable Distributed System Software.* IEEE Computer Society Press, Los Alamitos, CA, USA, 1985.

[96] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream control transmission protocol. RFC 2960, Oct. 2000.

[97] R. E. Tarjan. Testing graph connectivity. In *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing*, pp. 185–193, New York, NY, USA, 1974. ACM.

[98] The NetBSD Project. The NetBSD Project. <`http://www.netbsd.org/`>.

[99] The Zebra Project. GNU Zebra. <`http://www.zebra.org/`>.

[100] M. To and P. Neusy. Unavailability Analysis of Long-Haul Networks. *IEEE JSAC*, 12:100–09, Jan. 1994.

[101] L. Tong and K. S. Trivedi. An improved algorithm for coherent-system reliability. *IEEE Transaction on Reliability*, 47(1):73–78, 1998.

[102] UCLA Parallel Computing Laboratory. UCLA Parsec Programming Language. <`http://pcl.cs.ucla.edu/projects/parsec/`>.

[103] C. Villamizar, R. Chandra, and R. Govindan. BGP Route Flap Damping. RFC 2439, Nov. 1998.

[104] S. Vutukury and J. J. Garcia-Luna-Aceves. A simple approximation to minimum-delay routing. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 227–238, New York, NY, USA, 1999. ACM Press.

[105] S. Vutukury and J. J. Garcia-Luna-Aceves. An algorithm for multipath computation using distance-vectors with predecessor information. In *Proceedings of 8th International Conference on IEEE Computer Communications and Networks*, pp. 534–539, Oct 1999.

[106] S. Vutukury and J. J. Garcia-Luna-Aceves. MDVA: A Distance-Vector Multipath Routing Protocol. In *INFOCOM*, pp. 557–564, 2001.

[107] WIDE Project. WIDE PROJECT. <`http://www.wide.ad.jp/`>.

[108] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni. Traffic Engineering with MPLS in the Internet. *Network, IEEE*, 14(2):28–33, Mar/Apr 2000.

[109] X. Yang and D. Wetherall. Source selectable path diversity via routing deflections. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 159–170, New York, NY, USA, 2006. ACM Press.

[110] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *PADS '98: Proceedings of the twelfth workshop on Parallel and distributed simulation*, pp. 154–161, Washington, DC, USA, 1998. IEEE Computer Society.

[111] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 206–217, New York, NY, USA, 2003. ACM.

Research History

Publications

Journals

1. Yasuhiro Ohara, Hideaki Imaizumi, Akira Kato, Osamu Nakamura, and Jun Murai, "A Load Balancing Routing Algorithm for Wide Range of Traffic Demands", Transactions of Information Processing Society of Japan (IPSJ), Special Issue on User-centered Design and Management of Distributed Systems and the Internet, Vol. 48, No. 4, pp. 1627–1640 (Apr. 2007) (Japanese)

2. Yasuhiro Ohara, Hiroyuki Kusumoto, Osamu Nakamura, and Jun Murai, "Drouting Architecture: Improvement of Failure Avoidance Capability using Multipath Routing," The Institute of Electronics, Information and Communication Engineers (IEICE), 2008. (To be published)

3. Yasuhiro Ohara, Masaki Minami, Osamu Nakamura, and Jun Murai, "SimRouting: A Routing Simulation Tool," The Information Processing Society of Japan (IPSJ), 2008. (In submission)

International Conference

1. Yasuhiro Ohara, Manav Bhatia, Osamu Nakamura, and Jun Murai, "Route flapping effects on OSPF," In SAINT-03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops), pp. 232–237, Washington, DC, USA, IEEE Computer Society, 2003.

Miscellaneous

Software

- "Zebra ospf6d," <http://www.sfc.wide.ad.jp/~yasu/research/ospf-v3.html>

- “SimRouting,” <`http://www.simrouting.net/`>

Presentations at IETF

- “Status of OSPFv3 implementation for Zebra,” The 49th Internet Engineering Task Force, IPng working group, San Diego USA, dec 2000.
- “Implementation of OSPFv3 in Zebra,” The 48th Internet Engineering Task Force, OSPF working group, Pittsburgh USA, jul 2000.