



Julius-Maximilians-Universität Würzburg

Institut für Informatik

Lehrstuhl für Kommunikationsnetze

Prof. Dr.-Ing. P. Tran-Gia

Analysis and Optimization of Resilient Routing in Core Communication Networks

David Rogér Hock

Würzburger Beiträge zur
Leistungsbewertung Verteilter Systeme

Bericht 2/14

Würzburger Beiträge zur Leistungsbewertung Verteilter Systeme

Herausgeber

Prof. Dr.-Ing. P. Tran-Gia
Universität Würzburg
Institut für Informatik
Lehrstuhl für Kommunikationsnetze
Am Hubland
D-97074 Würzburg
Tel.: +49-931-31-86630
Fax.: +49-931-31-86632
email: trangia@informatik.uni-wuerzburg.de

Satz

Reproduktionsfähige Vorlage vom Autor.
Gesetzt in L^AT_EX Computer Modern 9pt.

ISSN 1432-8801

Analysis and Optimization of Resilient Routing in Core Communication Networks

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius–Maximilians–Universität Würzburg

vorgelegt von

David Rogér Hock

aus

Aschaffenburg

Würzburg 2014

Eingereicht am: 09.12.2013

bei der Fakultät für Mathematik und Informatik

1. Gutachter: Prof. Dr.-Ing. P. Tran-Gia

2. Gutachter: Prof. Dr. K. Tutschku

Tag der mündlichen Prüfung: 24.04.2014

Für Agnieszka, Jonathan und Raphael

Danksagung

Mit der Fertigstellung und Veröffentlichung dieser Doktorarbeit geht nun auch für mich ein Lebensabschnitt zu Ende. Wenn ich Ende September die Universität Würzburg verlasse, blicke ich auf etwas mehr als fünf Jahre als wissenschaftlicher Mitarbeiter am Lehrstuhl für Kommunikationsnetze zurück. Viele Leute haben mich auf diesem Weg begleitet und nun ist es an der Zeit, dafür Danke zu sagen.

Allen voran möchte ich mich ganz herzlich bei meinem Doktorvater, Herrn Prof. Dr.-Ing. Phuoc Tran-Gia, für die sehr gute Förderung, für die stets sehr angenehme Arbeitsatmosphäre am Lehrstuhl und für viele sehr fruchtbare wissenschaftliche Diskussionen bedanken. Herrn Prof. Tran-Gia ist es auch in ganz besonderem Maße zu verdanken, dass ich in vielen interessanten nationalen und internationalen Projekten mitarbeiten konnte und auch immer wieder Möglichkeiten hatte, an internationalen Tagungen und Konferenzen teilzunehmen, auf denen ich mich mit anderen Wissenschaftlern austauschen und meine Arbeiten diskutieren konnte. Schließlich sei hier noch erwähnt, dass ich auch über die reine wissenschaftliche Arbeit hinaus am Lehrstuhl viele abwechslungsreiche Aufgaben übernehmen durfte, bei denen ich sehr viel gelernt habe.

Als nächstes möchte ich meinem Zweitgutachter, Herrn Prof. Dr. Kurt Tutschku, für die bereitwillige Übernahme der Zweitkorrektur meiner Arbeit danken und auch für die wissenschaftlichen Diskussionen und die hilfreichen Kommentare im Vorfeld. Herrn Prof. Dr. Reiner Kolla und Herrn Prof. Dr. Andreas Hotho möchte ich für das Interesse an meiner Arbeit danken und dafür, dass sie als Prüfer für meine Disputation zur Verfügung standen.

Ebenfalls danken möchte ich Prof. Dr. Michal Pióro, Dr. Artur Tomaszewski und Cezary Żukowski, für die gemeinsamen Forschungsarbeiten im Rahmen des Projekts EuroNF. Diese Arbeiten haben es mir ermöglicht, einen weiteren Aspekt in meine Dissertation mit aufzunehmen.

Einen ganz besonderen Dank für die gute Zusammenarbeit möchte ich auch meinen früheren und aktuellen Kollegen aussprechen: Valentin Burger, Dr. Michael Duelli, Steffen Gebert, Matthias Hartmann, Dr. Robert Henjes, Matthias Hirth, Dr. Tobias Hoßfeld, Michael Jarschel, Dr. Dominik Klein, Prof. Dr. Alexander von Bodisco, Stanislav Lange, Dr. Frank Lehrieder, Prof. Dr. Michael Menth, Ngoc Anh Nguyen, Dr. Simon Oechsner, Dr. Rastin Pries, Marc Scheib, Dr. Daniel Schlosser, Christian Schwartz, Michael Seufert, Dr. Barbara Staehle, Dr. Dirk Staehle, Florian Wamser und Dr. Thomas Zinner. Durch die vielen, teils sehr angeregten fachlichen Diskussionen, die gegenseitige Hilfsbereitschaft und die vielen freundschaftlichen Unternehmungen auch außerhalb des Lehrstuhls haben sie sehr zur angenehmen Arbeitsatmosphäre und auch zum Gelingen meiner Arbeit beigetragen.

Besonders möchte ich an dieser Stelle auch noch einmal den früheren und aktuellen Gruppenleitern unseres Lehrstuhls danken: Dr. Tobias Hoßfeld, Prof. Dr. Michael Menth, Dr. Rastin Pries, Dr. Dirk Staehle und Dr. Thomas Zinner. Sie haben sich stets die Zeit für meine Anliegen genommen und standen immer mit Rat und Tat zur Seite. Ich habe dadurch nicht nur in Bezug auf das wissenschaftliche Arbeiten sehr viel gelernt.

Ich möchte mich auch bei allen Studenten bedanken, mit denen ich während der Zeit am Lehrstuhl zusammengearbeitet habe und die unsere Forschung und Projekte und teilweise durch Ihr Zuarbeiten auch das Gelingen dieser Doktorarbeit unterstützt haben. Ich möchte an dieser Stelle von einer namentlichen Nennung einzelner Personen absehen, schließe in meinen Dank aber alle studentischen Hilfskräfte, Diplomanden, Masteranden, Bacheloranden, Praktikanten, Seminaristen, usw. mit ein.

Weiterhin möchte ich mich auch bei unserer Sekretärin Frau Wichmann und ihrer Vorgängerin Frau Förster bedanken. Gerade bei den vielen anfallenden administrativen Aufgaben waren und sind Sie immer eine sehr große Hilfe!

Ich möchte mich natürlich vor allem auch bei meiner Familie und meinen Freunden bedanken, die immer an mich geglaubt und mir für den Abschluss der Promotion bis zum Ende die Daumen gedrückt haben. Besonders möchte ich mich bei meinen Eltern bedanken, dafür dass sie mich immer unterstützt und mir diese Ausbildung ermöglicht haben.

Last but not least, möchte ich mich bei meiner Frau Agnieszka und unseren zwei Kindern Jonathan und Raphael bedanken, denen ich auch diese Doktorarbeit widmen möchte. Gerade in den letzten Wochen und Monaten mussten sie mich oft mit der Unibibliothek oder meinem Laptop zuhause teilen und besonders in der Zeit direkt vor der Disputation war mir Agnieszka eine sehr große Hilfe.

Thank you very much, Dziękuję serdecznie & Vielen vielen Dank!

Würzburg, im August 2014

David Hock

Contents

1	Introduction	1
1.1	Scope	1
1.2	Scientific Contribution	4
1.3	Outline of This Thesis	7
2	Optimization of IP-based Resilient Routing	9
2.1	Fundamentals of IP Routing and Link Cost Optimization	12
2.1.1	Conventional IP Routing and Reconvergence	13
2.1.2	IP Fast Reroute	13
2.1.3	MPLS and MPLS Fast Reroute	14
2.1.4	Link Cost Optimization	16
2.2	Unique Shortest Paths: Prerequisite for Unambiguous Path Layouts	20
2.2.1	The Need for USP in IP Networks	21
2.2.2	Routing Optimization for USP	23
2.2.3	Performance Comparison	30
2.3	IP-based and Explicit Paths: The Cost of Better Performance . . .	35
2.3.1	Different Path Layouts for Primary and Backup LSPs . . .	35
2.3.2	Metrics to Study the Configuration Effort	38
2.3.3	Comparison of Differently Optimized Path Layouts . . .	42
2.4	Related Work	47
2.4.1	General Optimization of IP Routing	47
2.4.2	Tiebreakers and Unique Shortest Paths	48
2.4.3	Fast Reroute Mechanisms	49
2.5	Lessons Learned	50

3	Analysis of Optimization Effectiveness and Extension to the Full Failure Cycle	53
3.1	Effectiveness of Link Cost Optimization	56
3.1.1	Resilience Analysis	56
3.1.2	The ResiLyzer Tool	57
3.1.3	Results	62
3.2	Extension of the Optimization to the Full Failure Cycle	71
3.2.1	Failure Handling in Link State Routing and the Full Failure Cycle	71
3.2.2	Temporary Load Increase Caused by OFIB Orders	76
3.2.3	Analysis of the Maximum Relative Link Load during the OFIB Phase	79
3.2.4	Optimization of the Maximum Relative Link Load during the OFIB Phase	87
3.2.5	Optimization of the Full Failure Cycle	88
3.3	Related Work	93
3.3.1	Resilience and Resilience Analysis	93
3.3.2	Loop-Free Convergence and Ordered FIB Updates	94
3.4	Lessons Learned	96
4	Resilience Enhancements for the Control Plane of SDN-based Core Networks	101
4.1	Scenarios and Problem Description	104
4.1.1	SDN and SDN Scenarios	104
4.1.2	Resilient Controller Placement for SDN	106
4.2	Optimization of Resilient Controller Placement	110
4.2.1	Failure-Tolerant Controller Placement	110
4.2.2	Further Aspects of Resilient Placements	123
4.2.3	POCO-Framework	130
4.3	Related Work	134
4.4	Lessons Learned	137

5 Conclusion and Outlook	141
List of Acronyms	147
Nomenclature	151
Bibliography and References	155

1 Introduction

This monograph addresses the analysis and optimization of resilient routing in core communication networks. In the following, first a general introduction concerning the scope of this thesis is given. Then, the scientific contribution of this work is explained. Finally, an overview on the outline of this thesis is provided.

1.1 Scope

In recent years, the Internet has become more and more important in many different areas of daily life. One of the central tasks in the Internet is routing. Routing defines on which path packets are transmitted from the source of a connection to the destination. It allows to control the distribution of flows between different locations in the network and thereby is a means to influence the load distribution or to reach certain constraints imposed by particular applications. As failures in communication networks appear regularly [30] and cannot be completely avoided, routing is required to be resilient against such outages, i.e., routing still has to be able to forward packets on backup paths even if primary paths are not working any more. Consequently, not only the routing in the normal failure-free case is important, where all components are working correctly, but also the routing in case of outages. Furthermore, the constantly rising number of services with real-time requirements, such as, e.g., video streaming or Voice-over-IP [31], demand for a high resilience of routing and fast reaction in case of outages.

Throughout the years, various routing technologies have been introduced that are very different in their control structure, in their way of working, and in their ability to handle certain failure cases. Each of the different routing approaches

opens up their own specific questions regarding configuration, optimization, and inclusion of resilience issues. This work investigates, with the example of three particular routing technologies, some concrete issues regarding the analysis and optimization of resilience. Even though most of the results are analyzed and evaluated for particular technologies, the key concepts and methods used are not limited to that scope. In contrast, by analyzing specific issues regarding diverse routing approaches, the monograph contributes to a better general, technology-independent understanding of these approaches and of their diverse potential for the use in future network architectures.

Figure 1.1 gives an overview on the different technologies addressed in this work and on their resilience in different failure cases. In Figure 1.2, the same figure will be used as a base to illustrate the different contributions of this monograph.

The first considered routing type, is *decentralized intra-domain routing* based on administrative IP link costs and the shortest path principle. Typical examples are common today's intra-domain routing protocols *Open Shortest Path First (OSPF)* [32] and the *Intermediate System to Intermediate System Protocol (IS-IS)* [33]. This type of routing includes automatic restoration abilities in case of failures what makes it in general very robust even in the case of severe network outages including several failed components. Furthermore, special *IP Fast Reroute (IP-FRR)* mechanisms allow for a faster reaction on outages. The IP-FRR mechanism mainly regarded in this monograph are *Not-via addresses (NotVia)* whose current state has recently been published in RFC6981 [34]. For routing based on link costs, *Traffic Engineering (TE)*, e.g. the optimization of the maximum relative link load in the network, can be done indirectly by changing the administrative link costs to adequate values.

The second considered routing type, *Multiprotocol Label Switching (MPLS)-based routing* is based on the a priori configuration of primary and backup paths, so-called *Label Switched Paths (LSPs)*. Similar to IP-FRR, MPLS offers *MPLS Fast Reroute (MPLS-FRR)* options, namely one-to-one backup and facility backup [35]. The routing layout of MPLS paths offers more freedom compared

		Plain IP (e.g. OSPF, IS-IS)	MPLS	SDN (e.g. OpenFlow)
Control Approach		Decentralized routing computation in each node	Centralized computation and label (LSP) configuration	Typically centralized control plane realized by special controllers
	Forwarding	Based on destination address	Based on LSP label	Based on flow table entry
Resilient routing in different failure states Set of protected failure cases Failure-free case Severe, not protected failures (e.g., multiple failures)		Destination-based forwarding according to shortest path principle	Forwarding according to explicit preconfigured primary LSPs	Forwarding according to preconfigured flow table entries
	Short-term	Local rerouting using precalculated IP-FRR paths	Local rerouting using preconfigured backup MPLS-FRR LSPs	Local rerouting by requests of single forwarding nodes to their controllers
	Long-term	Autonomous global IP Restoration leading to a reconverged routing in the failure topology		Global rerouting by setting up new flow table entries in all forwarding nodes
		Routing possible as long as topology physically connected	If preconfigured LSPs not usable anymore, no routing possible	If controllers not reachable anymore, no rerouting possible

Figure 1.1: Overview on different routing technologies and their resilience.

to IP-based routing as it is not restricted by any shortest path constraints but any paths can be configured. However, due to its simplicity, a common approach is to base MPLS path layouts on the shortest paths in the underlying IP network. This can e.g. be done by means of *Label Distribution Protocol (LDP)* [36] or *Resource Reservation Protocol - Traffic Engineering (RSVP-TE)* [37]. In case of severe outages, when no preconfigured primary and backup paths can be used anymore, an MPLS based routing is not possible. In this case, e.g., a fallback on the underlying IP layer is necessary to enable a continuation of the routing.

Finally, in the third considered routing type, *typically centralized routing* using a *Software Defined Networking (SDN)* architecture, simple switches only forward packets according to routing decisions made by centralized controller units. The

most prominent example for such an architecture is OpenFlow [38]. SDN-based routing layouts offer the same freedom as for explicit paths configured using MPLS. In case of a failure, new rules can be setup by the controllers to continue the routing in the reduced topology. However, new resilience issues arise caused by the centralized architecture. If controllers are not reachable anymore, the forwarding rules in the single nodes cannot be adapted anymore. This might render a rerouting in case of connection problems in severe failure scenarios infeasible.

To address the contribution of this monograph, the following section summarizes the different resilience issues related to the considered routing types that are considered in this work.

1.2 Scientific Contribution

Figure 1.2 gives an overview on the different aspects and topics covered by this monograph as well as by research studies in related areas conducted by the author. Furthermore, the different tools are displayed that have been newly created or extended in course of the work presented here. In total, five issues are investigated and discussed in the monograph that will be briefly summarized in the following.

The TE for decentralized routing approaches based on link costs can be done only indirectly by setting adequate link costs. Depending on the configured link costs, several paths with identical path lengths might exist leading to an ambiguous path layout. In Chapter 2 of this monograph, possible problems arising with such ambiguous path layouts, especially considering also the IP-FRR case, are addressed. In particular, possible problems regarding the optimization of the maximum relative link load in the network are discussed and quantified for several example networks. A solution is presented that allows avoiding ambiguous path layouts by choosing adequate link costs during the optimization process without significantly worsening the quality of the optimization results in terms of maximum relative link load. In this context, the heuristic link cost optimization using the NetOpt tool is briefly introduced.

As indicated before, in case of MPLS based routing and, in particular, resilient MPLS routing with *Fast Reroute (FRR)* capabilities, there are more possibili-








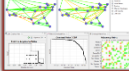

		Plain IP (e.g. OSPF, IS-IS)	MPLS	SDN (e.g. OpenFlow)
Control Approach		 NetOpt d routing ch node	Centralized computation and label (LSP) configuration	Typically centralized control plane realized by special controllers
Forwarding		 n address	Based on LSP label	Based on flow table entry
Resilient routing in different failure states Severe, not protected failures (e.g., multiple failures)	Short-term	Optimization of Unique Shortest Paths as prerequisite for unambiguous path layouts for failure-free case, reconverged case and IP-FRR case 	Comparison of IP-based and explicit path layouts regarding optimization potential and configuration effort 	Forwarding according to preconfigured flow table entries Local rerouting by requests of single forwarding nodes to their controllers 
	Long-term	Extension of the optimization to the Full Failure Cycle 	((1,11),[3,7,10,12,13,20]) Chapter 2	Global new fc ing up in all POCO
		Analysis of the effectiveness of the link cost optimization 	ResiLyzer LSPs not routing 	Resilience enhancements for the control plane of SDN-based core networks 
		((2,13,14,15), [25]) Chapter 3		((16) Chapter 4

Figure 1.2: Different aspects and topics covered by this monograph as well as by research studies in related areas conducted by the author. The content of references printed in bold is presented in this monograph.

ties for traffic engineering as the path layout is not restricted by any shortest path principle based on administrative link costs. However, this significantly increases the complexity of configuration of the paths, as each single path has to be setup in each *Label Switching Router (LSR)*. In the second part of Chapter 2 of this monograph, a comparison of MPLS path layouts based on IP shortest path routing and available explicit MPLS path layouts created by *Integer Linear Programs (ILPs)* [1, 12] without any layout restrictions is presented. The comparison considers both the primary paths in the failure-free case as well as MPLS-FRR backup paths. It is shown that there is a trade-off between configuration complexity and optimization potential for the maximum relative link load. The com-

parison helps network operators to decide which setup might be best for their particular need.

To keep the routing optimization process computationally feasible, e.g. when using heuristics for IP-based path layouts, only a limited set of the most probable failure scenarios, e.g. single link failures, can be examined. This yields the question whether the optimization for that limited set of failures performs well also for other possible failures. In Chapter 3 of this monograph, a framework for resilience analysis [39, 40] is used to visualize for an example network that optimization considering only single link failures also significantly improves the performance for scenarios with multiple simultaneous outages. In addition, the applied resilience analysis concept is briefly introduced and the functionality of the ResiLyzer tool implementing this concept is shown.

Another issue is addressed that is particularly important in the context of IP routing or MPLS routing based on IP shortest paths. IP routing adapts to the failure state and converges to a new routing layout automatically. However, as the new routing is calculated reactively to any kind of outage in a decentralized way, there is a period where old and new routing information might coexist in the network. This can lead to temporary forwarding loops, lost traffic and overloaded links. *Ordered FIB Updates (OFIB)* recently published in RFC6976 [41] are examined as one example for a so-called *Loop-Free Convergence (LFC)* concept and it is shown that this concept can be efficiently included into the IP routing optimization process. The work shows that including OFIB, it is possible to provide an optimization method for the full failure cycle of IP routing, i.e., including all routing stages, namely normal state, reconverged state, FRR state, and convergence phases.

Finally, in Chapter 4 of this work, particular questions arising with SDN are examined. The current movements towards SDN lead to the idea to implement traffic engineering by a centralized routing infrastructure consisting of simple forwarding switches and dedicated controllers providing routing decisions. This widens the range of possible TE options, as routing optimization can be realized on a per flow basis. Centralized routing via SDN offers the same freedom

in the routing layout as MPLS based paths layouts and can additionally react dynamically on outages in the network. However, despite of its new possibilities, SDN also opens up new resilience issues regarding the control plane. If a switch loses connection to its controller it is not able to change the routing anymore but will always stick to the currently configured forwarding rules. Furthermore, if a controller is overloaded by too many requests or the latency to the controller is too high, traffic engineering is exacerbated. Therefore, the particular issue of controller placement is investigated with regard to resilience and latency issues and it is shown that the resilience of an SDN architecture can significantly be improved when regarded during the placement decision. The *Pareto-based Optimal CONTroller-placement (POCO)* tool created to analyze the placements of controllers in SDN networks is presented.

1.3 Outline of This Thesis

Figure 1.3 provides an overview on the organization of this monograph. The issues mentioned before are presented in three chapters.

Chapter 2 addresses the optimization of IP-based resilient routing including *Unique Shortest Path (USP)* and a comparison of IP-based and explicit path layouts. Chapter 3 focuses on the analysis of the optimization effectiveness even when optimizing only for a small set of failures and on the extension of the optimization to consider the full failure cycle. In Chapter 4, resilience enhancements for the control plane of SDN-based core networks and the controller placement in SDN networks are discussed. Chapter 5 concludes the monograph.

The different issues addressed in this monograph involve different aspects and underlying technologies, such as routing optimization, resilience analysis, or controller placement. As mentioned before, different tools have been extended or newly created in course of the work presented here to address the discussed issues. Details to the considered technologies and tools as well as related work to the covered topics are provided in the single chapters in context of the corresponding issues.

Chapter 1: Introduction
Chapter 2: Optimization of IP-based Resilient Routing <ul style="list-style-type: none">• Fundamentals of IP Routing and Link Cost Optimization• Unique Shortest Paths: Prerequisite for Unambiguous Path Layouts• IP-based and Explicit Paths: The Cost of Better Performance• Related Work on IP Routing and Routing Optimization
Chapter 3: Analysis of Optimization Effectiveness and Extension to the Full Failure Cycle <ul style="list-style-type: none">• Effectiveness of Link Cost Optimization• Extension of the Optimization to the Full Failure Cycle• Related Work on Resilience Analysis and Loop Free Convergence
Chapter 4: Resilience Enhancements for the Control Plane of SDN-based Core Networks <ul style="list-style-type: none">• SDN Scenarios and Problem Description• Optimization of Resilient Controller Placement• Related Work on Controller Placement
Chapter 5: Conclusion and Outlook

Figure 1.3: *Organization of this monograph.*

2 Optimization of IP-based Resilient Routing

This chapter addresses the optimization of IP-based resilient routing. Independent of the underlying technology, basing the routing on the IP shortest path principle is very promising and a wide-spread practice due to its simplicity. In this case, routers forward packets along cost-minimal paths according to administrative link costs towards their destinations. Common today's intra-domain routing protocols, such as *Open Shortest Path First (OSPF)* [32] or the *Intermediate System to Intermediate System Protocol (IS-IS)* [33] directly determine the routing by administrative link costs. If *Multiprotocol Label Switching (MPLS)* is used, the path layout is in general not restricted by any shortest path principle but arbitrary *Label Switched Paths (LSPs)* could be configured. However, the LSP layouts can also be based on shortest paths in the underlying IP network, e.g., using the *Label Distribution Protocol (LDP)* [36] protocol. Depending on the particular use case, flow forwarding rules in *Software Defined Networking (SDN)* architectures as addressed in Chapter 4, can also be installed based on IP shortest paths.

One of the main advantages of the automatic routing calculation based on administrative link costs is the robustness against failures. When links or routers in an IP network fail, the information about the topology change is broadcast through the network and triggers the automatic recalculation of the forwarding tables in all routers. This IP reconvergence ensures that routers can again reach each other as long as the topology remains physically connected. One drawback of the reconvergence is that is slow and can take up to several seconds. To support the requirements of real-time applications, a faster reaction in case of outages is

required. Therefore, *Fast Reroute (FRR)* mechanisms have been developed that allow for a quick local reaction to outages. One promising FRR mechanism for IP networks offering a local protection for 100% of all single link and node failures are *Not-via addresses (NotVia)*. They have recently been published by the IETF as RFC6981 [34] and precalculate backup paths along shortest paths around a failed next-hop to the corresponding next-next-hop. For MPLS, there are two different FRR options both offering also 100% protection [35]. *Facility backup* uses local bypasses for backup traffic in a similar way as NotVia. *One-to-one backup* redirects traffic directly to its destination. As for the primary LSPs, if the paths are established along shortest IP paths, backup LSPs can be automatically set up and do not need to be configured with explicit paths. Further details and explanations to the FRR mechanisms will be given in Section 2.1.

Regarding the *Traffic Engineering (TE)* for IP-based routing, path layouts cannot be influenced directly but only indirectly by changing the administrative link costs to adequate values. Depending on the particular use case, there are different TE objectives [3, 10, 42]. The TE objective in this monograph is to minimize the maximum relative link load over all links in the network. When resilient routing is required, this also includes the relative link loads that occur in a certain set of protected failure scenarios. The optimization is done using heuristic link cost optimization based on threshold accepting. In this chapter, discussions of two issues in context of IP-based resilient optimization of the maximum relative link load are presented: i) ambiguous path layouts due to several paths with equal administrative path lengths, ii) a comparison of IP-based and explicit path layouts regarding optimization potential and configuration effort. Both issues will be briefly described in the following.

Depending on the configured administrative link costs, there often exist several paths with identical path lengths. Routers that use the *Equal-Cost Multi-Path (ECMP)* option distribute traffic over all available shortest paths. When *Single Shortest Path (SSP)* routing is required, each router uses a *Tie-Breaker (TB)* to select just one of the equal-cost shortest paths. However, TBs are not properly standardized and might even use non-deterministic information like, e.g., router-

internal interface numbers whose order can change after restart or the current link load, or even select a random next hop. This makes the paths of general SSP routing hard to predict. Traffic engineering techniques like routing optimization or admission control need to know which links carry which traffic. Therefore, link cost settings are preferred that lead to a *Unique Shortest Path (USP)* routing without equal-cost paths so that no TBs are required. The same problem of several equal cost paths appears for backup paths based on administrative link costs. To predict the impact of backup traffic for resource management, the backup paths must be unique in order to be predictable. Therefore, the underlying IP routing should produce only unique backup paths.

Two main questions regarding USP are analyzed: i) do USP layouts exist for all networks also including the backup path layouts for different failures and ii) if yes, can USP layouts be found that are of an equal quality in terms of maximum relative link load as regular SSP routing layouts?

In this chapter, it is first shown that optimized SSP routing can lead to significantly higher load than expected when TB decisions differ from the assumptions made during the optimization. This cannot happen when the routing produces USP in all protected scenarios what however puts more constraints on the link cost settings. A heuristic algorithm is used to find link cost settings that result in USP routing, i.e., a routing layout not involving any ambiguous equal-cost paths. The link cost settings are optimized in order to minimize the maximum relative link load for a set of considered failure scenarios. It is illustrated that the fraction of available USP routings as well as the quality of the optimized routings depend on the maximum allowed link costs. The heuristic is adapted to produce optimized USP link cost settings for the FRR mechanisms mentioned above. Finally, the performance of optimized ECMP, SSP, USP routing in failure-free IP networks as well as for IP reconvergence and various FRR mechanisms are compared for a limited set of protected failure cases.

The second issue discussed in this chapter addresses the restrictions to the path layout imposed by TE based on administrative link costs. As mentioned before, in case of MPLS based routing and, in particular, resilient MPLS routing with

FRR capabilities, there are more possibilities for traffic engineering as the path layout is not restricted by any shortest path principle based on administrative link costs. However, the resulting path layouts can significantly increase the complexity of configuration of the paths, as each single path has to be setup in each *Label Switching Router (LSR)*. A comparison of the optimized MPLS path layouts based on IP shortest path routing and explicit MPLS path layouts created by colleagues from the Technical University of Warsaw using *Mixed Integer Linear Programs (MILPs)* without any layout restrictions [1, 12] is presented. This helps to answer two particular questions: 1) how large is the optimization potential for the maximum relative link load when explicit path layouts are used and ii) what is the additional configuration complexity imposed by such layouts? The comparison helps network operators to decide which setup might be best for their particular need.

The main part of this chapter is taken from [11]. The explanation of the fundamentals of IP routing and link cost optimization in Section 2.1 is mainly aggregated from [11, 13]. The comparison of IP-based and explicit path layout summarizes previous work [1, 12] which has been done in cooperation with researchers from the Technical University of Warsaw. The remainder of this chapter is as follows. Section 2.1 gives an overview on the fundamentals of IP routing and link cost optimization. Section 2.2 discusses the problems of ambiguous path layouts due to several equal-cost paths and addresses the optimization of USP routing. Section 2.3 provides a comparison of routing layouts based on administrative link costs and explicit layouts regarding optimization potential and configuration effort. Section 2.4 gives an overview on related work. The chapter is concluded and results are summarized in Section 2.5.

2.1 Fundamentals of IP Routing and Link Cost Optimization

This section briefly explains the fundamentals of IP routing and link cost optimization. First, conventional IP routing, reconvergence, as well as *IP Fast*

Reroute (IP-FRR) and *MPLS Fast Reroute (MPLS-FRR)* are introduced. Then, link cost optimization is addressed.

2.1.1 Conventional IP Routing and Reconvergence

In intra-domain IP networks, routers exchange information about the topology and administrative link costs with each other. Based on these routing messages, each node obtains a full view of the link topology including administrative link costs. It uses this information to set up the routing table whereby it associates any destination in the network with the interface leading towards a least-cost path to the destination. Thus, the routing table helps to look up onto which outgoing interface packets destined to a certain node in the network should be forwarded.

In case of a modification of the topology, e.g., due to a link or router failure, a reconvergence process is invoked. The change is broadcast through the entire local network and routers recalculate the outgoing interface mapping in their routing tables based on the new topology. As long as the network is physically connected, IP routing finds new routes for all source-destination pairs. This makes it very robust against network failures.

2.1.2 IP Fast Reroute

The reconvergence process in IP networks can take up to several minutes. During this time, forwarding loops can appear when some of the routers have updated their routing tables earlier than others. As a consequence, the affected traffic cannot be delivered to its destination. Further, looping the traffic causes high load on the respective links which causes additional overload. To avoid this phenomenon, IP-FRR has been proposed. Routers detecting a failure immediately switch the affected traffic to preestablished backup paths that are likely to be unaffected by the observed failure. There are multiple proposals for the implementation of IP-FRR, such as, e.g., *Loop-Free Alternates (LFAs)* [43], *Remote LFAs* [44], *Maximally Redundant Trees (MRTs)* [45, 46], and *Multiple Routing*

Configurations (MRC) [47]. Overviews on the different proposals can be found, e.g., in [48–50] and the references within.

The FRR approach considered in the following are NotVia addresses which have recently been published as an RFC by the *Internet Engineering Task Force (IETF)* [34, 51]. For any node N , there is a NotVia address N_F and packets addressed to N_F are forwarded to N while node F is avoided on the path. Hence, the routing tables in the network require additional entries for these NotVia addresses. They are used for IP-FRR as indicated in Figure 2.1 and explained in the following. It is assumed that a node A receives a packet that is normally forwarded over F and the *Next-Next-Hop (NNHOP)* N to its destination, but the *Next Hop (NHOP)* F has failed, see Figure 2.1(a). Then the node A encapsulates this packet towards the NotVia address N_F to tunnel it to N . N decapsulates the packet and forwards it to the destination. If the NHOP F is already the destination, see Figure 2.1(b), the packet can be delivered if only the link from A to F is down but not F itself. Then, A encapsulates the packet to F_A and forwards it to some of its neighbor nodes so that the packet is carried towards F avoiding the link from A to F . Hence, the NotVia mechanism leads the traffic on the shortest path according to administrative link costs around the NHOP to the NNHOP or around the next-link to the NHOP if the NHOP is the destination node. If due to an additional network failure, traffic encapsulated with a NotVia address is tunneled again, this can lead to traffic loops in the network. To avoid this problem, already NotVia encapsulated traffic must not be tunneled to NotVia addresses again but be dropped instead.

2.1.3 MPLS and MPLS Fast Reroute

As mentioned before, in MPLS networks, routing is realized on base of LSPs. Each packet is assigned a label and based on this label the packet is forwarded according to the corresponding LSP. LSPs can be set up either using pre-computed, explicit paths or along the shortest paths in an underlying IP network. If the second option is considered, to set up an LSP, signaling packets are forwarded

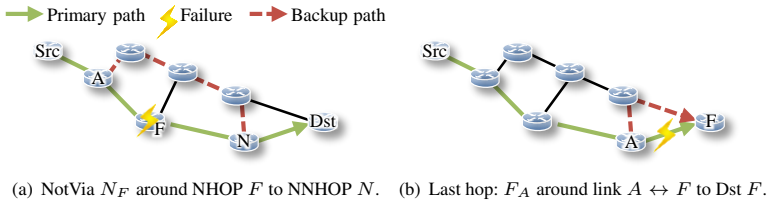


Figure 2.1: *NotVia* backup paths.

from the ingress router to the egress router of the LSP and, thereby, determine its path layout. To allow for a fast reaction in case of failures, MPLS as well comes with FRR capabilities similar to IP-FRR. There are two MPLS-FRR options. The structure of their backup paths is briefly described in the following. Further information on MPLS, MPLS-FRR, and resilient mechanisms in general can also be found in [52, 53].

Facility Backup

The facility backup option bypasses traffic around a failed network element. These bypasses are established from the *Point of Local Repair (PLR)* where the failure is detected along a shortest path around the failure towards the *Merge Point (MP)* where they rejoin the primary LSP. Figure 2.2 illustrates the two available bypass options for link and node failures, *LinkBypass(PLR, MP)* and *NodeBypass(PLR, MP)*, respectively. In case of *NodeBypass(PLR, MP)*, when the MP is the NNHOP after the failure, *NotVia* and facility backup use the same backup path layout.

One-to-One Backup

Instead of a single bypass for all LSPs, one-to-one backup creates an individual backup path for each flow. These paths are established along shortest

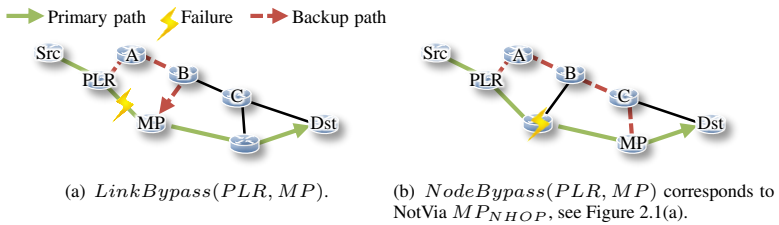


Figure 2.2: Facility backup uses bypass tunnels.

paths from the PLR directly to the egress router of the LSP, which leads to a different path layout than facility backup. Figure 2.3 illustrates the two available detour options for link and node failures, $LinkDetour(PLR, Dst)$ and $NodeDetour(PLR, Dst)$, respectively.

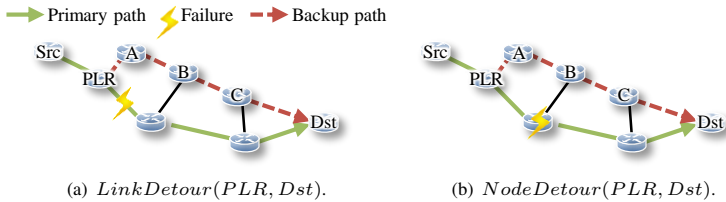


Figure 2.3: One-to-one backup uses detour tunnels.

2.1.4 Link Cost Optimization

IP routing follows the least-cost paths according to administrative link costs. Traffic engineering is possible by link cost optimization, i.e., by appropriately choosing those link costs that lead to a good load distribution on the links. An

objective function defines what is understood by a good load distribution and is later discussed in more detail.

Optimization Algorithm

The input for link cost optimization are a network topology, link capacities, a traffic matrix, and a given set of so-called protected failure scenarios \mathcal{X} for which the routing should be optimized. The output of the process are administrative costs for all links in the network. The set \mathcal{X} usually comprises all single link and/or node failures (\mathcal{L} , \mathcal{N} , \mathcal{LN}). The failure-free state $s = \emptyset$ is always part of this set.

Finding optimum link costs for a given objective function is usually an NP-hard problem even when only considering the failure-free case \emptyset . Therefore, heuristic algorithms are used to search good link costs. Here, as a base for the later extensions, the heuristic optimizer NetOpt from previous work [10, 54] implemented in Java is used. Figure 2.4 shows a screenshot of NetOpt displaying the same network and routing as depicted in Figure 2.1(a). The illustration indicates that the outage of a node automatically also involves the outage of the adjacent links. NetOpt works on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with routers \mathcal{V} and links \mathcal{E} , and minimizes a given optimization objective ψ . Every link $e \in \mathcal{E}$ is assigned an administrative cost value $k \in [1 : k_{\max}]$ and thus, the search space consists of $(k_{\max})^{|\mathcal{E}|}$ different link cost settings. The optimizer implements a threshold accepting heuristic. It starts with a random link cost configuration \mathbf{k} , and uses neighborhoods where up to 25% of the link weights are randomly changed. When a new neighbor link cost configuration is better or not worse than a threshold above the current best value, it is accepted as new current configuration and the search continues. If no strictly better solution is found after a previously configured number of iterations, the currently best value is returned as the final optimization value. The heuristic can be restarted several times with different random start configurations \mathbf{k} and the best link cost setting of all runs is returned as final result.

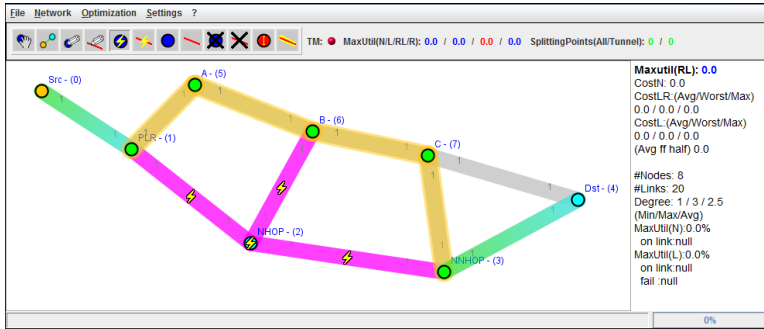


Figure 2.4: Screenshot of the NetOpt GUI.

Objective functions

In [10], different objective functions for resilient and non-resilient IP routing were studied, which can be used for different application scenarios. Two of them are explained in more detail here. Both take the relative link load as a parameter. The relative load $\rho(l)$ of a link l is calculated as the quotient of the total traffic on a link and the link's capacity. To illustrate the severeness of possible overload, relative link loads larger than 100% are allowed in the computation.

- $\rho_{\mathcal{X}}^{\max}$ is the maximum relative link load of all links in all protected failure scenarios \mathcal{X} . It is a good choice, if routing optimization is used to guarantee that certain constraints on the relative link load are kept. It can be calculated as

$$\rho_{\mathcal{X}}^{\max} = \max_{(s \in \mathcal{X})} \max_{(l \in \mathcal{E})} \rho(l). \quad (2.1)$$

- $\phi_{\mathcal{X}}^{\text{weighted}}$ sums up penalties over all links and all protected failure scenarios whereby these penalties increase with increasing relative link load. The

penalties are calculated with Fortz’s continuous, piecewise linear, monotonically increasing penalty function ϕ [55], which is illustrated in Figure 2.5. According to Fortz et al. the idea of the increasing penalties is that it is much cheaper to transport traffic on low loaded links and links get more sensitive when they are highly loaded. Furthermore, in particular load values close to or above 100% should be avoided. This can be addressed by the increasing penalty values. The objective function $\phi_{\mathcal{X}}^{\text{weighted}}$ is good if the main focus of the optimization lies on the overall link loads and the average path lengths. A more formal definition is given in [10].

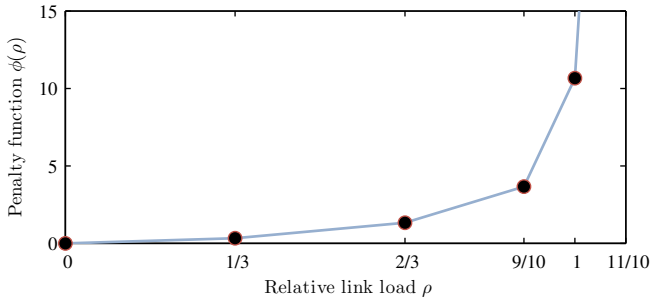


Figure 2.5: Fortz’s load-dependent penalty function ϕ .

Different objective functions lead to significantly different optimization results. To visualize that, routing based on the *Hop-Count (HC)* metric and optimized routing based on objective functions $\rho_{\mathcal{X}}^{\text{max}}$ and $\phi_{\mathcal{X}}^{\text{weighted}}$ is considered whereby \mathcal{X} comprises all single link failures

Figure 2.6 shows the maximum relative link load of all links in the COST239 network [10] in all protected failure scenarios \mathcal{L} . The x-axis indicates the relative link load and the y-axis the fraction of links whose maximum link load exceeds the value on the x-axis. HC routing leads to the highest values, optimized routing based on $\rho_{\mathcal{X}}^{\text{max}}$ leads to the lowest values. Objective function $\phi_{\mathcal{X}}^{\text{weighted}}$ achieves

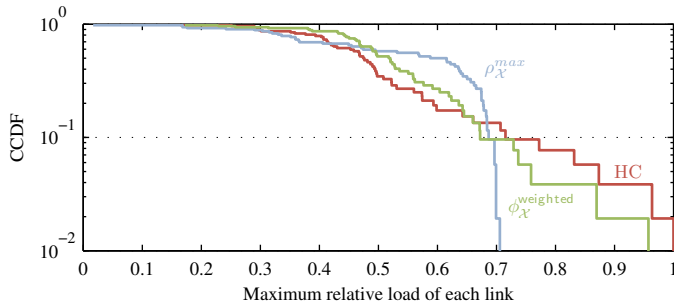


Figure 2.6: CCDF of the maximum relative link load over all single link failure scenarios (COST239 network).

a compromise. The drawback of ρ_{χ}^{\max} is that only the relative link load of the most loaded link is considered. Thus, optimization using this objective function does not improve the second-worst link when the worst link cannot be improved further as this would not lead to any improvement of the ρ_{χ}^{\max} value. Therefore, in course of [10], NetOpt was extended to allow for the combined optimization of two objective functions. If first ρ_{χ}^{\max} is minimized and then $\phi_{\chi}^{\text{weighted}}$, this leads to the lowest maximum relative link loads and reduces also the load on other highly loaded links.

2.2 Unique Shortest Paths: Prerequisite for Unambiguous Path Layouts

This section discusses the issue of *Unique Shortest Path (USP)*. First, the need for USP in IP networks is motivated. Then, the impact of wrong TB assumptions is quantified, extensions to NetOpt to optimize USP routing are described, and some examples discussed. Finally, the routing optimization in NetOpt is extended towards FRR mechanisms and the performance of various routing and resilience mechanisms is compared.

2.2.1 The Need for USP in IP Networks

In the following, some scenarios where USP routing is desired are discussed.

Classic IP Networks

Shortest path routing in IP networks is unpredictable when equal-cost paths exist. Using ECMP, traffic is equally distributed over all shortest paths. The load balancing is done on a per-flow basis. For each flow from any source to any destination, a hash value is calculated that is used to decide on which of the equal cost paths the flow is going to be forwarded. As the sizes of different flows vary, an equal distribution of the traffic is subject to statistical variations. Further details regarding this load balancing and the statistical variations can be found in [56–58]. The prospective path of a new flow is also unpredictable which leads to problems with path-dependent flow control functions like admission control and flow termination [59]. SSP does not have these issues because only a single path for each source-destination pair is used. However, the use of non-deterministic TB decisions leads to an unpredictable path layout. This can be avoided by using USP. Figures 2.7(a) and 2.7(b) illustrate example cases where there are multiple shortest paths between source and destination if all links are considered to have equal costs 1. By increasing the administrative link costs of one of the two equal cost paths, a USP layout can be obtained.

NotVia IP Fast Reroute

In the NotVia IP-FRR mechanism, additional NotVia addresses are used to reach the NNHOP towards the destination not via the failed router or link. Thus, each router is not only reachable by its regular address but also with several NotVia addresses corresponding to the failures of the different direct neighbors. The routes to these NotVia addresses are pre-computed by all routers in a distributed way based on the normal IP link costs but without using the indicated NotVia-router. Then, they are stored in the forwarding tables of the routers. A comprehensive description of the mechanism can be found in [48].

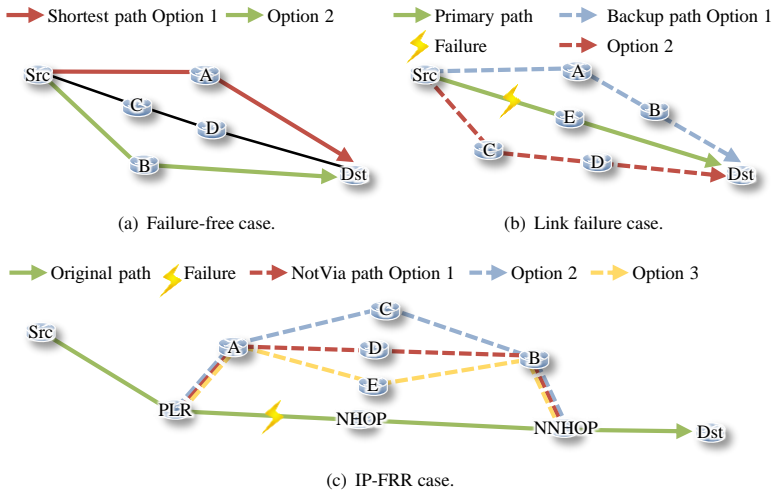


Figure 2.7: Illustration of the multiple shortest path problem.

When SSP routing is used and several equal-cost shortest paths around a failed NHOP exist, the layout of the NotVia backup paths is hard to predict. Figure 2.7(c) shows a part of a network illustrating an example for that situation. In case of the failure of the link between PLR and NHOP, there are three equal cost NotVia paths to the NNHOP. Unlike in classic IP networks, in this case, ECMP cannot be used to reach a load balancing on all of these paths. Since all rerouted packets are encapsulated, they appear to come from the PLR and are addressed to the NNHOP after the failure. Therefore, the hash-based load balancing algorithms, e.g. in node *A* treat them like a single flow, i.e., they are all carried over one single backup path. Hence, there is no way to balance the backup traffic over equal-cost paths and the chosen backup path is also hard to predict if the details of the load balancing algorithms are not known. Thus, for IP-FRR using NotVia, USP routing is even more desired than in classic IP networks.

MPLS Fast Reroute (MPLS-FRR)

In MPLS networks, when LSPs are setup along the shortest paths in an underlying IP network, signaling packets are forwarded from the ingress router to the egress router of the LSPs and, thereby, determine the path layout. When equal-cost paths exist, it is almost impossible to predict the exact path layout. As this holds for both SSP and ECMP routing, USP routing is a desired feature also in IP-based MPLS networks.

2.2.2 Routing Optimization for USP

For optimized SSP routing, the impact of unknown TBs on the relative link load is quantified. It is shown that the maximum link costs must be large enough so that link cost settings with USP routing can be efficiently found and effectively optimized. Finally, the optimal link cost ranges also for the optimization of other routing mechanisms are determined.

Networks under Study

For the experiments, the Rocketfuel topologies [60] are used in addition to the well known Cost239 [61] and Geant [62] networks. All nodes that are connected by only a single link are removed because they are never used to forward other traffic and, thus, are irrelevant for traffic engineering. The sizes and the node degrees of the networks are compiled in Table 2.1. To generate synthetic traffic matrices resembling real-world data the method proposed in [63] and enhanced in [64] is used. The traffic matrices are scaled so that the maximum relative link load over all single link failures ρ_{λ}^{\max} equals 100% for ECMP routing based on the HC metric, i.e. all link costs are set to 1. For the comparisons, theoretical relative link loads above 100% are allowed without packet drops.

Table 2.1: Networks under study

ID	Name	$ \mathcal{V} $	$ \mathcal{E} $	<i>AvgDeg</i>
AB	Abovenet	20	156	7.8
AT	AT&T	28	120	4.28
CO	Cost239	11	52	4.73
EB	Ebone	25	126	5.04
EX	Exodus	22	102	4.64
GE	Geant	19	60	3.16
SP	Sprintlink	33	190	5.78
TI	Tiscali	38	232	6.11

Impact of Non-Deterministic Tie-Breaker

Link cost optimization is an offline process, i.e., an external tool takes a representation of the network and its traffic matrix as input and returns an optimized link cost setting. These link costs are then configured in the real network. For SSP routing, the tools have to assume certain TBs though the routers in the network might use different ones. When link costs are optimized for wrong TBs, routers possibly send traffic to already highly loaded links. This has a devastating effect on the relative link load. This problem is quantified in the following.

First, SSP routing is optimized and the lowest NHOP port number is used as TB. An optimized link cost setting \mathbf{k}_0 is obtained that results in a maximum relative link load $\rho_{\mathcal{X},0}^{\max}$. This link cost setting \mathbf{k}_0 is applied to the network. Then the port-numbers are randomly jumbled in each router which leads to different TB decisions and $\rho_{\mathcal{X},1}^{\max}$ for the new routing is calculated. This experiment is repeated 100 times and the worst values $\rho_{\mathcal{X},\text{worst}}^{\max} = \max_{(i=1..100)}(\rho_{\mathcal{X},i}^{\max})$ for each network are recorded. The experiment is conducted both under failure-free conditions, $\mathcal{X} = \emptyset$, and including single link failures, $\mathcal{X} = \mathcal{L}$. The results for each network are presented in Figure 2.8, which shows the possible increase in maximum relative link load $(\frac{\rho_{\mathcal{X},\text{worst}}^{\max}}{\rho_{\mathcal{X},0}^{\max}} - 1)$ in percent. Different TBs can significantly increase the maximum relative link load both under failure-free conditions and

in case of single link failures. For example in the Tiscali network with failure-free routing, the relative link load is over 180% higher. The presented results can differ for other initial (optimized) link cost settings \mathbf{k}_0 , and many TB changes that affect only slightly loaded paths have no effect at all on the maximum relative link load. Therefore, these results cannot be generalized, but they show that traffic engineering is useless when relying on unknown TBs.

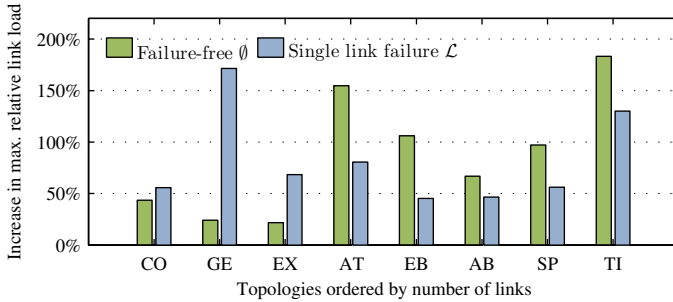
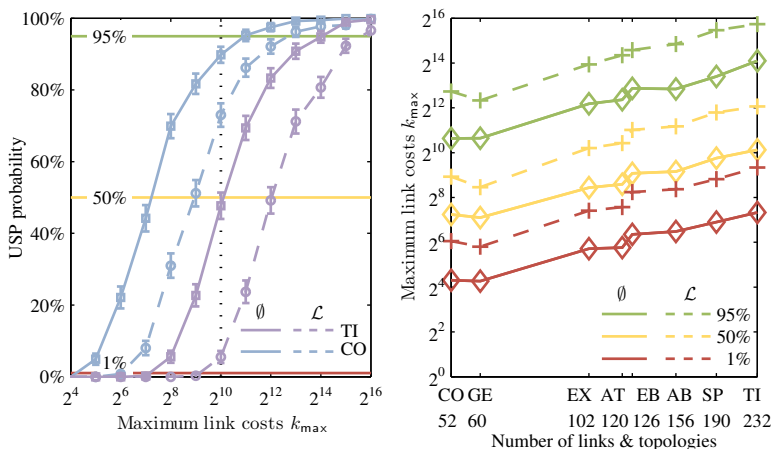


Figure 2.8: Possible increase of max. relative link load with different TBs

USP Probability

The allowed maximum link cost k_{\max} constrains optimized path layout. When k_{\max} is small, equal-cost paths cannot be avoided whereas with a larger link cost range, the routing can be configured so that all split points can be eliminated. To illustrate this, the USP probability for different k_{\max} values is empirically quantified, i.e., the probability that a random routing configuration with link costs $k \in [1 : k_{\max}]$ has only unique shortest paths. For each evaluated k_{\max} , 1000 random link cost configurations \mathbf{k} are generated and it is counted how many of these settings produce USP routing. Figure 2.9(a) shows the resulting estimated USP probabilities in the smallest (Cost239) and the largest (Tiscali) evaluated network both for failure-free and resilient routing. The 99% confidence intervals

are indicated to show the accuracy of the results. The USP probability is close to zero for small k_{\max} . At a certain value it clearly increases and approaches 100% when k_{\max} is large enough. For resilient routing, the USP probability is much smaller than for failure-free routing. This is due to the fact that resilient routing requires USPs for all protected failure scenarios \mathcal{X} while failure-free routing requires USPs only for the failure-free case \emptyset . Hence, any link cost setting producing USPs for resilient routing produces USPs also for failure-free routing but not vice-versa. The differences in USP probability can be rather large. For example in the Tiscali network with $k_{\max} = 2^{10}$, indicated by a vertical grid line, the USP probability is below 10% for resilient routing (dashed line) and already around 50% for failure-free USP routing (solid line).



(a) USP probabilities for selected networks (b) Maximum link costs for which the USP probability is 1%, 50%, and 95%.

Figure 2.9: Dependence of the USP probability on the maximum link cost and the network size.

The shape of the USP probability curves is almost identical for all evaluated network topologies but their positions on the x-axis differ. Figure 2.9(b) describes the location of the curve for all investigated networks. It shows the k_{\max} values for which the USP probability is approximately 1%, 50%, and 95% on the logarithmic y-axis. These probabilities are also indicated in Figure 2.9(a). The topologies are placed on the logarithmic x-axis according to their number of links. For better readability, the values of the different topologies are connected. The lines have an approximately linear shape, i.e., due to the logarithmic scale on both axes, k_{\max} seems to grow polynomially with the number of links in the network, but in fact k_{\max} also highly depends on the structure of the topology. The curves for 1%, 50%, and 95% USP probability are far apart from each other. That means, to clearly increase the USP probability, the maximum link cost k_{\max} needs to be significantly increased. This holds for any network size. To obtain the same USP probability for resilient routing as for failure-free routing, the maximum link cost settings need to be about four times larger.

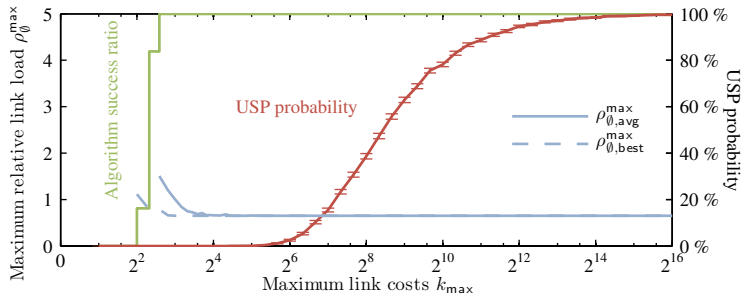
Routing Optimization for USP

It is easy to manually create a USP routing for failure-free conditions. Simply, all links on a spanning tree in the network are set to link cost 1 and all other links get a very high value so that only links on the tree are used for (USP-)routing. Another option which even works during any failure scenario is to number the links consecutively and assign a link cost of 2^i to the link with the number i . In this case, it is impossible to get the same cost sum over different paths and the USP property is ensured. The latter method only works for networks with up to 16 links. Both methods lead to very bad load balancing and high link loads as only few links are used. A good USP routing distributes traffic so that the maximum relative link load is low. This can be achieved by routing optimization. The link cost optimization heuristic NetOpt presented before is extended to generate USP routings and it is then shown under which circumstances good results are received.

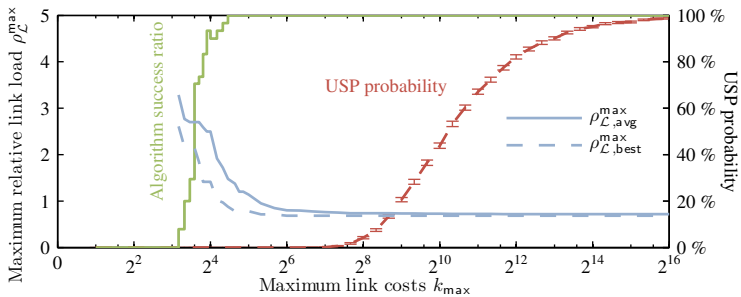
Extensions to NetOpt For USP routing, equal-cost paths must not exist. Inspired by [65], the objective function $\rho_{\mathcal{X}}^{\max}$ is extended and each equal-cost path split is penalized with a high penalty value 50. Good $\rho_{\mathcal{X}}^{\max}$ values should be smaller than 1.0 (100% relative link load) and even bad configurations can never lead to (theoretical) link loads higher than by the factor of 50. Therefore, this penalty term ensures that even the worst USP solution is preferred to any non-USP solution. Using this simple extension, NetOpt searches systematically for USP routings. However, it might not find a USP solution at all if k_{\max} was not chosen large enough.

Impact of Link Cost Ranges on Max. Relative Link Load The routing in the Exodus network is optimized for different k_{\max} values. For each k_{\max} , 50 optimizations are performed for failure-free, $\mathcal{X} = \emptyset$, and for resilient routing, $\mathcal{X} = \mathcal{L}$. Figure 2.10 shows the best and the average maximum relative link load values of the 50 experiments, $\rho_{\mathcal{X},\text{best}}^{\max}$ and $\rho_{\mathcal{X},\text{avg}}^{\max}$, respectively. Because the runtime of the heuristic is limited, the obtained results are not necessarily optimal. The values are connected with a solid line for better readability. The success ratio in the figure indicates the percentage of optimizations where NetOpt could find a USP solution, see right y-axis. For very small costs, no solutions were found. For $k_{\max} \in [4 : 7]$ in failure-free routing (Figure 2.10(a)) and for $k_{\max} \in [9 : 20]$ in resilient routing (Figure 2.10(b)), NetOpt is more and more successful in finding USP solutions. With larger allowed costs, it always finds a USP routing. For small k_{\max} , the only possible USP routings that are found lead to a high maximum relative link load. Good relative link loads could be achieved only with higher maximum link costs k_{\max} . Analog to the success ratio, resilient USP routing requires a higher k_{\max} than failure-free routing to achieve good results. The figure also shows for the different k_{\max} values the previously addressed empirical USP probability determined by evaluating 1000 random link cost settings from Section 2.2.2. Surprisingly, good results are already found in cost ranges where the USP probability is still smaller than 0.1%.

2.2 Unique Shortest Paths: Prerequisite for Unambiguous Path Layouts



(a) Failure-free scenario: $\mathcal{X} = \emptyset$.



(b) Failure-free and all single link failure scenarios: $\mathcal{X} = \mathcal{L}$.

Figure 2.10: Success ratio of NetOpt and maximum relative link load of optimized link cost settings depending on the maximum link costs k_{\max} in the Exodus network.

To determine the best k_{\max} for different routing mechanisms, the maximum relative link loads of resilient routing, $\mathcal{X} = \mathcal{L}$, are optimized for ECMP, SSP, and USP routing. Figure 2.11 presents the best and average results, $\rho_{\mathcal{X},\text{best}}^{\max}$ and $\rho_{\mathcal{X},\text{avg}}^{\max}$, out of 50 optimizations for each k_{\max} in the Cost239 network. For USP, the average and best relative link load values decrease with increasing k_{\max} . For SSP, the average and best quality of the optimized routing solutions are almost independent of the maximum link costs. For ECMP, the best link cost settings are found

for small k_{\max} and the quality of the resulting routing deteriorates with increasing maximum link costs. At first sight, this is surprising since all routings with small maximum link cost can equally be configured with a larger k_{\max} . However, larger maximum link costs increase the search space so that the heuristic optimization using NetOpt cannot find the best equal-cost path solutions anymore. The described phenomena are very similar for all examined network topologies, and are also true for the FRR mechanisms. Therefore, for the optimizations in [11], a small $k_{\max} = 8$ were used for ECMP and SSP and a large $k_{\max} = 2^{16}$ for USP.

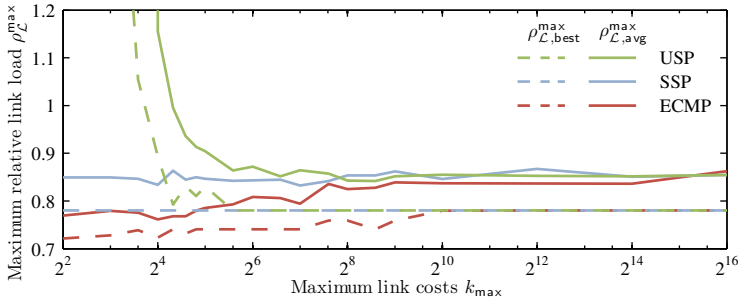


Figure 2.11: Maximum relative link load for optimized resilient routing depending on the maximum link costs k_{\max} .

2.2.3 Performance Comparison

It was shown that with extensions to NetOpt, USP layouts can be efficiently found. The quality of these layouts regarding the maximum relative link load $\rho_{\mathcal{X}}^{\max}$ depends on the maximum allowed link costs k_{\max} . In the following, maximum relative link load $\rho_{\mathcal{X}}^{\max}$ and average path length for optimized IP routing are studied in more detail. First, the performance of ECMP, SSP, and USP routing under failure-free conditions, $\mathcal{X} = \emptyset$, is investigated. Then, resilient routing, $\mathcal{X} = \mathcal{L}$, is regarded and IP routing reconvergence based on ECMP, SSP, and USP is compared with the FRR schemes NotVia and MPLS one-to-one backup. For each of the various routing options, in [11], 50 optimization runs were per-

formed and the link cost setting with the lowest maximum relative link load $\rho_{\mathcal{X}}^{\max}$ were selected for the analysis.

In course of different extensions to NetOpt following [11], e.g., in [3, 15] a database of all optimization results and corresponding link cost vectors for all considered networks was created. Due to different combinations of two objective functions, different run times and different thresholds, for some topologies improved values compared to the values regarded in [11] could be found. The scope of the rest of this chapter is on the quality of the best results rather than on the way of their optimization. Therefore, if not stated else wise, from now on, for each network and each considered objective the best value out of all optimized values is considered independent of the used parameter settings to obtain this result.

Optimized Routing under Failure-Free Conditions

The performance of optimized USP routing under failure-free conditions is studied and it is compared with optimized ECMP and SSP. Figure 2.12(a) shows the maximum relative link load ρ_{\emptyset}^{\max} for several networks relative to the one of un-optimized ECMP HC routing. For almost all reported networks, ECMP clearly achieves the lowest maximum relative link load, and SSP and USP routing mostly lead to higher relative link load values¹. While the actual performance of the different routing options strongly depends on the network topology, USP routing is almost always as good as regular SSP routing with a slight difference for the SP and TI network. Figure 2.12(b) shows the average path length of the resulting routings, normalized by the average path length of HC routing which produces shortest paths.

Formally, the average path length in the failure-free case $\pi_{\emptyset}^{\text{avg path length}}$ is defined as the average number of hops for all paths $p_{v,w}^{\emptyset} \subseteq \mathcal{E}$ between any nodes

¹By definition, the best ECMP and SSP values are always at least as good as the best USP values. This is due to the fact that each USP routing is in particular also a valid ECMP and SSP routing. As USP routings are fulfilling the USP constraints, they obviously lead to the same routing layout and maximum relative link load for ECMP and SSP.

$v, w \in \mathcal{V}$ in the failure-free case:

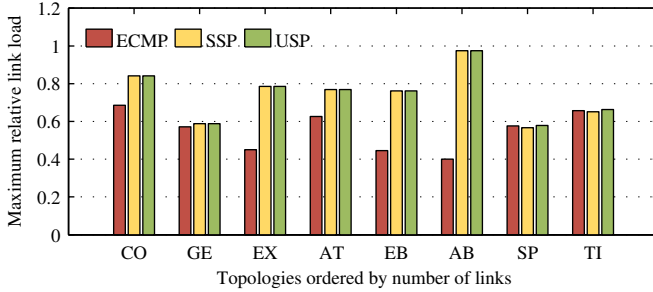
$$\pi_{\emptyset}^{\text{avg path length}} = \frac{\sum_{(v,w \in \mathcal{V})} |p_{v,w}^{\emptyset}|}{|\mathcal{V}| \cdot (|\mathcal{V}| - 1)}. \quad (2.2)$$

In case of ECMP, if there are several paths for a demand from v to w , for the calculation, $p_{v,w}^{\emptyset}$ is set to the one of these paths with the longest hop count. The path lengths for SSP and USP routing hardly differ. The ECMP paths are often slightly longer because in the calculation only the longest partial path is included, and routing optimization artificially prolongs some partial paths by assigning them the same costs as to shorter paths in order to balance traffic over all of them.

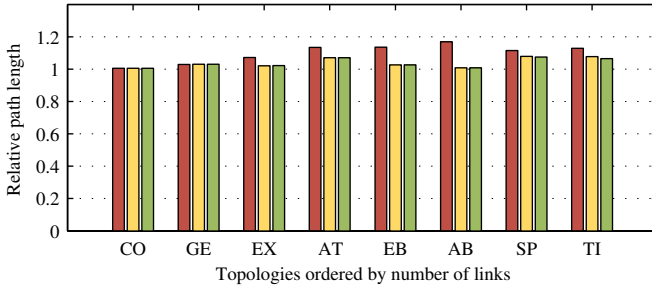
Optimized Resilient IP Routing including Fast Reroute

To study the performance of resilient routing, the link costs are optimized to minimize the maximum relative link load over all single link failure scenarios \mathcal{L} . The ECMP, SSP, and USP routing options are investigated with reconvergence in failure cases, as well as the NotVia IP-FRR mechanism which has the same path layout as the facility backup method for MPLS-FRR, and the one-to-one backup method for MPLS-FRR.

Figure 2.13(a) shows the maximum relative link load of all links in all considered failure scenarios \mathcal{L} relative to the one of ECMP HC routing. The maximum relative link load is again in almost all networks the same for SSP and USP and only slightly differs in the AT network. This holds also when FRR mechanisms are used. Thus the SSP FRR results are omitted in Figure 2.13. ECMP leads to the lowest maximum relative link load and the differences to USP are smaller than under failure-free conditions. FRR mechanisms mostly cause larger maximum relative link loads because rerouted traffic continues to be carried close to the outage location which experiences an increased load of backup traffic. In contrast, IP reconvergence mechanisms can reroute the traffic more evenly through the network.



(a) Maximum relative link load relative to HC routing.



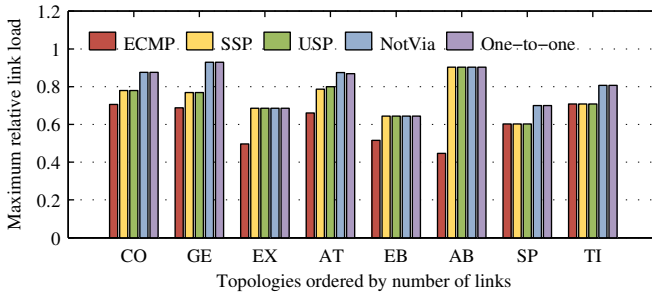
(b) Average path length relative to HC routing.

Figure 2.12: Comparison of optimized ECMP, SSP, and USP routing under failure-free conditions.

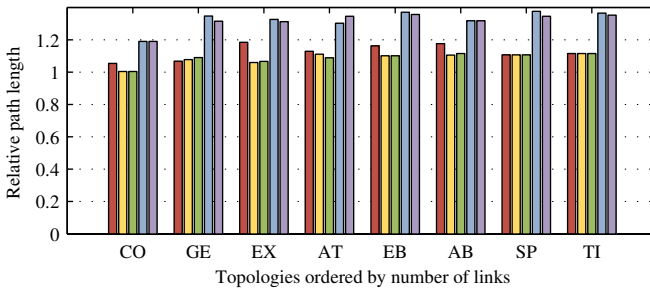
Figure 2.13(b) compares the path length prolongation of the different routing mechanisms compared to HC routing. For each ingress-egress aggregate (v, w) with $v, w \in \mathcal{V}$, the longest path $p_{v,w}^s \subseteq \mathcal{E}$ that occurs in any failure scenario $s \in \mathcal{L}$ is taken and the average length of these longest paths is calculated:

$$\pi_{\mathcal{L}}^{\text{avg path length}} = \frac{\sum_{(v,w \in \mathcal{V})} \max_{(s \in \mathcal{L})} |p_{v,w}^s|}{|\mathcal{V}| \cdot (|\mathcal{V}| - 1)}. \quad (2.3)$$

As before, for ECMP, only the longest partial path of all equal cost paths is considered in the calculation. The path lengths for ECMP, SSP, and USP are quite similar. The FRR mechanisms prolong the backup paths dramatically because traffic is forwarded on detours around the failure and not on the second-best paths starting at the source as in normal IP reconvergence. NotVia (MPLS facility backup) usually has the longest paths because it takes a bypass to the NNHOP router instead of tunneling directly to the destination like MPLS one-to-one backup.



(a) Maximum relative link load relative to HC routing.



(b) Maximum path length relative to HC routing.

Figure 2.13: Comparison of optimized ECMP, SSP, and USP routing as well as FRR mechanisms under all single link failures \mathcal{L}

2.3 IP-based and Explicit Paths: The Cost of Better Performance

In the previous section, it was shown that ambiguous path layouts can be efficiently avoided when including USP constraints in the link cost optimization. In the following, a comparison between IP-based and explicit MPLS path layouts is addressed. As described before, MPLS and MPLS-FRR basically offer the possibility to configure any layout using so called LSPs. Compared to layouts based on IP shortest paths, however, this can lead to a significant increase in configuration effort. In the following, with the example of MPLS including the MPLS one-to-one backup option, the potential of optimizing the maximum relative link load as well the configuration effort are compared and quantified for the networks discussed before in course of USP. First, different path layouts, explicit ones and those based on IP shortest paths, are explained. Then, the metrics used for the numerical comparison are introduced. Finally, numerical results from the comparison are presented.

2.3.1 Different Path Layouts for Primary and Backup LSPs

MPLS and MPLS-FRR path layouts can either be configured explicitly to allow for any arbitrary layout or automatically based on the underlying IP plane. The way of configuration gives different constraints to the layout and also influences the different optimization possibilities. In the following, the different possible layouts are illustrated and their optimization is briefly addressed.

Figures 2.14 and 2.15 illustrate path layouts based on IP-based USP and explicit paths respectively. As the main intention is to show the general concept of the different routing layouts, all examples are only showing layouts of primary paths, i.e., routing in the failure-free case \emptyset . Backup paths (e.g., those using the MPLS one-to-one backup option) follow the same restrictions and are therefore not displayed here.

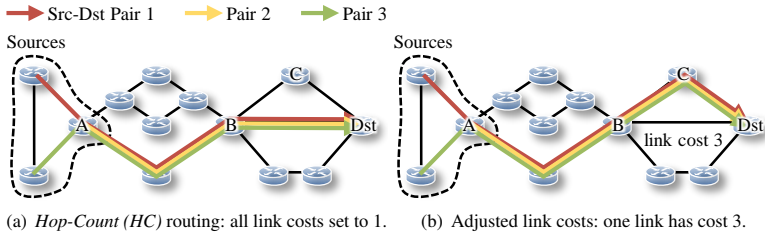


Figure 2.14: Illustration of LSP layouts based on IP link costs.

In Figure 2.14, two different path layouts based on IP link costs are depicted. If not stated differently, all links in the displayed topology are supposed to have equal link cost of 1. Shortest path routing based on administrative link costs is destination based. That means that independent of the source of a given demand all packets destined to the same destination are forwarded to the same *Next Hop (NHOP)*². An example is given in Figure 2.14(a), where all flows originating from different sources are sent to the *Dst* node on the same shortest path starting from node *A*. IP routing layouts can not be optimized directly, but only by choosing adequate link costs. An example for adjusted link costs is shown in Figure 2.14(b). When the cost of the link between nodes *B* and *Dst* are changed to 3 still having all other links with cost 1, the displayed shortest path from the sources to the destination are moved to the links from *B* to *C* and from *C* to *Dst*. However, all flows are moved to the same new path due to the destination based routing layout.

To be able to split the different demands to the same destination onto several different paths (with possibly different IP shortest path length) explicit path layouts have to be defined. An example for such a layout is displayed in Fig-

²An exception to that behavior might be ECMP layouts as described before where traffic is split onto paths of equal length on a per flow (source-destination) base. However, as multiple paths are not possible in case of FRR the ECMP option is not further regarded here.

Figure 2.15(a). Explicit MPLS paths allow for source routing. That means that by adequately labeling packets at the source router the path of a packet can be assigned to preconfigured LSPs and thus arbitrary layouts can be defined by previously installing LSP forwarding rules at the routers.

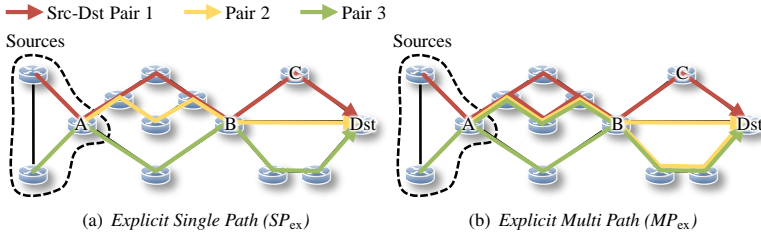


Figure 2.15: Illustration of LSP layouts based on explicit path layouts.

The layout displayed is a single path layout, i.e., between each source and destination there is exactly one path configured and the demand is not split into any subpaths. In the following, this layout is described as SP_{ex} . To indicate more clearly that the previous USP layouts are based on IP link costs, in the following, they are referred to as *IP-based USP* (USP_{IP}). As Figure 2.15(a) indicates, different SP_{ex} LSPs can partly share common nodes or links on their paths, but they can spread again into separate paths at any node in the graph. Obviously, such path layouts increase the possibilities for routing optimization compared to link cost optimized USP_{IP} . However, the key to reach better routing layouts might be the prolongation of certain paths to reach a better load balancing. The longer paths lengths though lead to a higher configuration effort as (1) the LSPs need to be configured in more routers, (2) the longer a primary path is, the more backup paths are necessary to protect all links and nodes of the path. This again increases the number of necessary LSPs for the backup paths.

Figure 2.15(b) shows a generalization of SP_{ex} , in the following called *Explicit Multi Path* (MP_{ex}). In this type of routing, flows do not need to be routed on

a single path but can be arbitrarily split onto several paths between the source and the destination. MP_{ex} layouts further increase the possibilities during routing optimization as an even better load balancing is possible than when calculating an SP_{ex} layout. However, this advantage tends to come with the cost of a higher configuration effort. Each split of a flow onto several subpaths however involves the definition of additional LSPs. Intermediate nodes are only forwarding traffic according to the labeled LSP but are not able to perform any autonomous splitting of the traffic. Therefore, for each possible path from the source to the destination an individual LSP has to be configured starting from the source node and the source node can then split the flow on the different LSPs³. This situation is even more exacerbated as each primary LSP needs to be individually protected by a backup LSP.

The optimization of explicit routing layouts is not restricted by as many constraints as IP based shortest path routing. Therefore, it fits especially well for being optimized with means of MILPs. In [1, 12], mathematical formulations were presented to obtain for MPLS including the MPLS one-to-one backup option, SP_{ex} and MP_{ex} layouts for the topologies regarded before in the context of USP. In the following, a comparison of these layouts with the best IP based layouts considered in the previous section is presented. Regarding the restrictions made to the different routing layouts, the possible USP_{IP} layouts are a real subset of the possible SP_{ex} layouts that are again a real subset of the MP_{ex} layouts ($USP_{IP} \subset SP_{ex} \subset MP_{ex}$). Therefore, two questions are investigated in the following: i) how does the removal of certain layout restrictions increase the optimization potential regarding the maximum relative link load and ii) what increase in configuration effort does that involve?

2.3.2 Metrics to Study the Configuration Effort

In the following, the metrics used to compare the different layouts concerning their configuration effort are introduced. The main configuration effort in the con-

³A more detailed definition and illustration of this metric is illustrated in the following section.

text of MPLS consists of the setup of the LSP labels in the nodes of the network. Therefore, informally spoken, the configuration effort increases with the number of different labels in general and the number of labels per node more particular. The number of LSP labels that need to be installed again depends on the number of paths in the network, i.e., the number of primary paths for the failure-free case as well as the number of backup paths in case of outages. The number of backup paths increases with the length of the primary paths as each element on the primary path can possibly fail and such failures have to be protected. In this section, the failure-free case as well as all single link failures are regarded. That means that from the MPLS one-to-one backup option the *LinkDetour*(PLR, Dst) is regarded.

To allow for a quantification of the configuration effort in terms of number of paths or labels, in the following subsections, two metrics are introduced. Both are illustrated for the failure-free case in Figure 2.16.

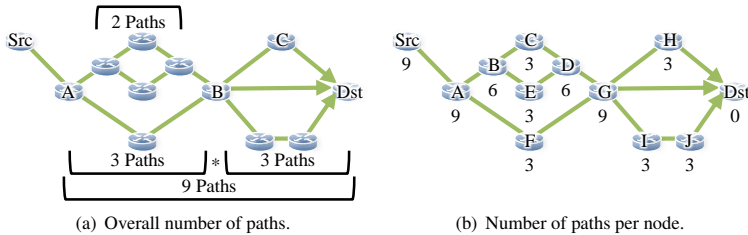


Figure 2.16: Illustration of different metrics to study the configuration effort.

Overall number of paths

The first considered metric is the overall number of paths in the entire network. The concept how this metric is calculated is depicted in Figure 2.16(a). In case of single paths, i.e., USP_{IP} and SP_{ex} , there is by design exactly one primary path per demand. Considering the backup paths for single link failures, for each demand, there is exactly one (single) backup path for the failure of each link on

the primary path of that demand. The backup path always reaches from the PLR corresponding to the failed link to the destination of the demand.⁴ The metric is more complex for the case of multi path routing, i.e., MP_{ex} . Whenever a demand is split into several subdemands at any hop of a path, this split has to be invoked already at the source of the demand by assigning adequate labels. That means that at each splitting point of a demand the number of paths leading to that point and the number of paths leaving that point have to be multiplied to compute the total number of necessary LSPs to realize all possible combinations. This calculation is illustrated in Figure 2.16(a) for a demand splitting the paths on all links from node Src to node Dst . There are three possible paths from the source Src to reach node B and another three paths from node B to reach the destination Dst . To allow for sending packets on any arbitrary combination of the first three subpaths and the second three subpaths, in total $3 \cdot 3 = 9$ LSPs have to be configured. As mentioned before, only the failure-free case is displayed here. The complexity of the metric further increases when regarding backup paths. For each possible PLR on each configured primary LSP (nine in this case) backup LSPs have to be considered. These backup paths can again be arbitrarily split on multiple paths increasing the overall number of LSPs⁵.

Formally, the overall number of paths is defined as follows. Let $LSP_{v,w}^{primary}$ be the set of all primary paths p between nodes $v, w \in \mathcal{V}$ as explained before. Let further each path $p \subseteq \mathcal{E}$ consist of a set of links l that are part of that path. Then, for the outage of each link l of a path p there is a set of backup paths $LSP_{v,w,p,l}^{backup}$ that protect the path p between nodes v and w for that particular failure. Backup paths start at the PLR and end at the destination of the demand, i.e., $LinkDetour(PLR, Dst)$. The overall number of paths for a demand between nodes v and w is then calculated as

$$|LSP_{v,w}^{overall}| = |LSP_{v,w}^{primary}| + \sum_{p \in LSP_{v,w}^{primary}} \sum_{l \in p} |LSP_{v,w,p,l}^{backup}|. \quad (2.4)$$

⁴Technically, several parallel paths from the same PLR to the same destination for different demands might be aggregated. However, this demands additional case studies and thereby further increases the configuration effort. Therefore, here no aggregation of parallel paths is considered.

⁵For the same reasons as explained before, here, a merging of several parallel paths is not considered.

For the special case of single path routing layouts USP_{IP} and SP_{ex} as described above, there is only a single primary path $p_{v,w}^{\emptyset} \in LSP_{v,w}^{primary}$. The metric can then be simplified to

$$|LSP_{v,w}^{overall}| = 1 + |p_{v,w}^{\emptyset}|. \quad (2.5)$$

The overall number of paths for the entire network is calculated as

$$|LSP^{overall}| = \sum_{v,w \in \mathcal{V}} |LSP_{v,w}^{overall}|. \quad (2.6)$$

Number of paths per node

To get a more precise measure of the effort related with the configuration of single nodes, as a second metric the (average and maximum) number of paths per node is considered. Formally, $LSP_{v,w}^{primary,n} \subset LSP_{v,w}^{primary}$ is defined as the set of all primary paths p between nodes $v, w \in \mathcal{V}$ that pass through node n . Furthermore, the set of all incoming paths of a node n is defined as $-LSP^{primary,n} \subset LSP^{primary,n}$ and contains all primary path through n between arbitrary nodes v and w with $v \neq n$. That means that paths starting at node n are not counted as incoming. Analogously, the outgoing paths of node n , $+LSP^{primary,n} \subset LSP^{primary,n}$, are defined as all those paths through n not ending at n . The sets of incoming and outgoing backup paths, $-LSP^{backup,n}$ and $+LSP^{backup,n}$, are defined analogously. In the following discussion, only the outgoing paths are regarded. For each (primary and backup) path leaving a node, the node needs a forwarding rule for the corresponding label. Thus, the number of outgoing LSPs equals the number of MPLS forwarding rules in a node. Therefore, this seems to be a good measure for the configuration effort. An example is illustrated in Figure 2.16(b). The source node has a total of 9 outgoing LSPs since all paths, as computed in Figure 2.16(a), are leaving the source node. Analogously, the destination node has no outgoing but only incoming paths. In all other nodes of the network the number of incoming paths from source to desti-

nation equals the number of outgoing paths ${}^+LSP^{\text{primary},n} = -LSP^{\text{primary},n} = LSP^{\text{primary},n}$.

As metric for the comparison in the following section, the average and maximum number of (outgoing) paths per node, $|{}^+LSP_{\text{avg}}^{\text{overall}}|$ and $|{}^+LSP_{\text{max}}^{\text{overall}}|$, are considered. These are calculated respectively as follows:

$$|{}^+LSP_{\text{avg}}^{\text{overall}}| = \frac{\sum_{n \in \mathcal{V}} (|{}^+LSP^{\text{primary},n}| + |{}^+LSP^{\text{backup},n}|)}{|\mathcal{V}|}, \quad (2.7)$$

$$|{}^+LSP_{\text{max}}^{\text{overall}}| = \max_{n \in \mathcal{V}} (|{}^+LSP^{\text{primary},n}| + |{}^+LSP^{\text{backup},n}|). \quad (2.8)$$

2.3.3 Comparison of Differently Optimized Path Layouts

In this section, the differently optimized layouts, USP_{IP} , SP_{ex} , and MP_{ex} are compared regarding the potential to optimize the maximum relative link load and the configuration effort. The same networks are investigated as before in the context of USP, see Table 2.1. Additionally, the Labnet (LA) [66] network is considered due to its special properties. It has a total of 20 nodes as the Abovenet network but only 53 links compared to 156 in the Abovenet network. Therefore, different to Abovenet, the routing in the Labnet topology can be solved optimally with the mathematical formulations of [1, 12] in reasonable computation time. Table 2.2 compiles for the regarded networks the values of $\rho_{\mathcal{L}}^{\text{max}}$, $|LSP^{\text{primary}}|$, and $|LSP^{\text{backup}}|$ for the different layouts. These values will be addressed subsequently in the next paragraphs. As mentioned before, the USP_{IP} routing layouts correspond to those shown before in Section 2.2. The layouts MP_{ex} and SP_{ex} have been obtained in course of the work [1, 12]. For the upper three networks in Table 2.2, CO, GE, and LA, the SP_{ex} and MP_{ex} values correspond to the optimal possible solutions. For all other networks, in the lower part of the table, the values are not optimal due to too long computation time of the used MILPs. Instead,

2.3 IP-based and Explicit Paths: The Cost of Better Performance

these values correspond to the best values reachable after a computation limited to at most seven days per network.

Table 2.2: Performance metrics for optimized primary and backup path layouts using IP-based USP (USP_{IP}), Explicit Single Path (SP_{ex}), and Explicit Multi Path (MP_{ex}).

Net- work Id	Max. relative link load $\rho_{\mathcal{L}}^{\max}$ in %			Number of primary paths $ LSP^{\text{primary}} $			Number of backup paths $ LSP^{\text{backup}} $		
	USP_{IP}	SP_{ex}	MP_{ex}	USP_{IP}	SP_{ex}	MP_{ex}	USP_{IP}	SP_{ex}	MP_{ex}
CO	87.60	83.74	64.19	110	110	127	172	226	324
GE	92.93	79.15	71.64	342	342	355	870	958	1005
LA	67.61	45.36	38.79	380	380	483	837	1012	1928
AB	90.31	90.65	22.99	380	380	479	683	865	2917
AT	86.90	73.40	47.78	756	756	803	2050	2233	3013
EB	64.34	65.37	30.92	600	600	690	1335	1586	2614
EX	68.52	66.63	33.35	462	462	538	991	1156	2033
SP	69.99	65.33	53.71	1056	1056	1127	2567	3679	4183
TI	80.74	79.22	71.73	1406	1406	1422	3105	4214	4259
Avg	186%	172%	100%	100%	100%	113%	100%	124%	201%

Comparison of the Maximum Relative Link Load

First, the maximum relative link load for the different layouts is investigated. Normally, SP_{ex} routing layouts are expected to yield better or at least equal $\rho_{\mathcal{L}}^{\max}$ values as USP_{IP} , as SP_{ex} allows less constrained layouts and in particular all USP_{IP} layouts can also be configured as SP_{ex} layouts. It can be seen for most networks that this expectation holds. However, the results also show that the quantitative difference between the $\rho_{\mathcal{L}}^{\max}$ values of both routing layouts strongly depends on the network. In two of the regarded networks, AB and EB, the $\rho_{\mathcal{L}}^{\max}$ value resulting from the USP_{IP} layout is actually slightly better than the $\rho_{\mathcal{L}}^{\max}$ resulting from the SP_{ex} layout. This is of course not intuitive due to the relation between SP_{ex}

and USP_{IP} mentioned before. It shows however that the SP_{ex} layouts from [1, 12] presented here are not optimal but they only represent some intermediate result after interruption of the MILP. A comparison of the SP_{ex} and MP_{ex} shows that the removal of the single path constraint can in some networks significantly increase the optimization potential and lead to much better maximum relative link load values. However, again this strongly depends on the considered network. To quantify the relation of the $\rho_{\mathcal{L}}^{\max}$ values of the different layouts, the last row shows the average $\rho_{\mathcal{L}}^{\max}$ value of the layouts, USP_{IP} and SP_{ex} , compared to the value of MP_{ex} as reference. On average, SP_{ex} leads to an about 72% higher maximum relative link load. USP_{IP} on average increases this value by another $(186/172) - 100\% = 8\%$.

Comparison of the Configuration Effort

Next the configuration effort is compared for the different layouts. Therefore, subsequently the two different metrics introduced before are looked at.

Comparison of the Overall Number of Paths A comparison of the different layouts concerning the overall number of paths is displayed at the right part of Table 2.2. Both, the number of primary paths and the number of backup paths, $|LSP^{\text{primary}}|$ and $|LSP^{\text{backup}}|$, are compared. By design, for the primary paths, both single path layouts, USP_{IP} and SP_{ex} , have an equal number of one path per demand, i.e. $|LSP^{\text{primary}}| = |\mathcal{V}| \cdot (|\mathcal{V}| - 1)$. The number of backup paths for the single path layouts corresponds as mentioned before in Equation 2.5 to the sum of the path lengths of all primary paths, $\sum_{p \in LSP^{\text{primary}}} |p|$, as each link of each primary path has exactly one backup path. Obviously, SP_{ex} tends to accept longer path lengths in order to reach better $\rho_{\mathcal{L}}^{\max}$ values. Therefore, SP_{ex} has more backup paths than USP_{IP} . On average, see last row of the table, it has 24% more backup paths. An interesting, yet due to the prior explanations not surprising finding is, that in particular, also in the AB and EB networks, where the considered USP_{IP} layout leads to better $\rho_{\mathcal{L}}^{\max}$ values than the SP_{ex} layout, still, more backup paths

exist for the SP_{ex} layout than for the USP_{IP} layout. The number of paths of the MP_{ex} layouts is much higher than for the single path layouts. This is already the case when considering only primary paths and can be explained by the illustration shown before in Figure 2.16(a). Each split has to be considered and increases the overall number of LSPs. In the failure case, when backup paths are considered, the number of paths for MP_{ex} increases even more drastically compared to the primary paths, as more primary paths with possible longer path lengths lead to a large increase of the possible outages that need to be protected with backup paths. In average, when looking at the last row of Table 2.2, MP_{ex} layouts have about twice as many backup paths as USP_{IP} . This shows that a choice has to be conducted by network operators, what is preferred for the particular use case: better optimization potential in terms of $\rho_{\mathcal{L}}^{\max}$ or less configuration effort in terms of the number of LSPs.

Comparison of the Average and Maximum Number of Paths per Node

As second metric for configuration effort, the average and maximum number of paths per node is considered. Table 2.3 shows the metrics $|{}^+LSP_{avg}^{overall}|$ and $|{}^+LSP_{max}^{overall}|$ for all discussed routing layouts.

The qualitative results of the paths per node basically correspond to the results of the overall paths in the network. MP_{ex} leads to the largest number of overall paths and also to the largest average and maximum number of paths per node. USP_{IP} leads to the lowest number of paths per node and overall paths. Quantitatively, however, the difference between the discussed layouts considering the number of paths per node is exacerbated. This is due to the fact that the number of paths per node does not only depend on the overall number of paths in the network but also on the average path lengths of these paths. The longer a path is, the more nodes it passes and the more nodes need to be configured with the corresponding LSP label. Thus, e.g., for the single path layouts USP_{IP} and SP_{ex} , even though the number of primary paths is identical for both layouts the average and maximum number of primary paths per node (not shown in the table) is higher for SP_{ex} due to longer average path lengths. Table 2.3 shows only the total number

Table 2.3: Average and maximum number of outgoing paths per node.

Network ID	Number of paths per node ($ ^+LSP_{avg}^{overall} / ^+LSP_{max}^{overall} $)		
	USP _{IP}	SP _{ex}	MP _{ex}
CO	49.45 / 65	81.55 / 150	113.18 / 200
GE	202.58 / 612	259.74 / 736	274.16 / 764
LA	149.20 / 327	234.90 / 459	403.35 / 822
AB	107.75 / 228	151.70 / 419	532.30 / 1418
AT	290.00 / 829	393.29 / 1153	538.00 / 1665
EB	199.08 / 608	315.44 / 900	497.16 / 1367
EX	166.73 / 553	225.46 / 778	428.68 / 1330
SP	313.94 / 664	579.33 / 1674	654.82 / 1920
TI	302.42 / 1223	500.37 / 2751	504.79 / 2757
Avg	100% / 100%	152% / 176%	244% / 276%

of paths, i.e. the sum of primary and backup paths, where the difference is even larger. A particularly interesting finding that is revealed by the table concerns the MP_{ex} layouts. Obviously, the number of paths per node for MP_{ex} is, depending on the network, much higher than for SP_{ex} or USP_{IP}. However, in the CO network, the maximum number of paths per node rises up to 200. The overall number of paths for CO, see Table 2.2, is only $127 + 324 = 451$. That means that there is at least one node in the CO network that for the optimal MP_{ex} layout is part of more than 40% of all LSPs.

Before concluding this section, the results are summarized by an illustration in Figure 2.17. The figure visualizes the relative difference of the discussed layouts regarding the different considered metrics $\rho_{\mathcal{L}}^{\max}$, $|LSP^{\text{backup}}|$, $|^+LSP_{avg}^{overall}|$, and $|^+LSP_{max}^{overall}|$. The values correspond to those printed in bold in the last rows of Tables 2.2 and 2.3.

The graph shows that single paths layouts based on explicit paths, SP_{ex}, allow for a better maximum relative link load optimization than layouts constrained by IP shortest paths, USP_{IP}. However, on average this difference is only around 8%.

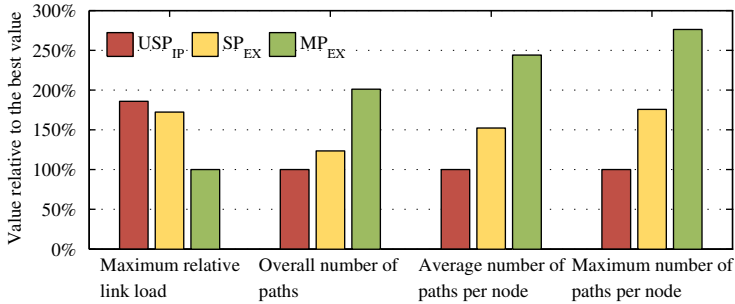


Figure 2.17: Comparison of the different path layouts regarding different metrics.

Still, SP_{ex} has a much higher configuration effort compared to USP_{IP} . If path layouts including multiple paths are allowed, the optimization potential of the maximum relative link load can be significantly improved. However, this leads to a significant increase of configuration complexity even compared to explicit single paths.

2.4 Related Work

This section presents a brief review on existing work regarding the optimization of IP routing in general, the existing work on USP routing, and FRR mechanisms.

2.4.1 General Optimization of IP Routing

A review on different objective functions used for the optimization of IP routing is provided in [10]. The maximum relative load of all links [67–72] and the continuous, piece-wise linear, monotonically increasing penalty function proposed by Fortz et al. [55, 72–75] are most frequently used. In [75] the authors propose an optimization of the link weights by iteratively modifying the slopes of

the Fortz objective function. Fortz et al. were the first to consider reactions to changed networking conditions like sudden congestion due to increased user activity or backup traffic after rerouting. They proposed to modify only a few link costs to adapt the routing to the new conditions [76, 77].

Objective functions can also take a limited set of "protected" failure situations into account. In case of failures, IP routing reconverges which leads to changed load conditions. Optimization of resilient routing attempts to improve routing also under failure conditions using the same set of link cost settings. It has been first presented in [78–80]. Later contributions look at faster heuristics [81] or alternative objective functions for special application scenarios [54, 82].

Optimizing IP routing is an NP-complete problem even in the failure-free case [83–85]. It can be optimally solved by (Integer, Mixed Integer) Linear Programs (ILPs, MILPs, LPs) [70, 83, 86–93]. As already mentioned during the discussion of Section 2.3, due to the large solution space, these methods are mostly applicable only to small networks because of long runtimes. Therefore, faster heuristics are frequently used. Local search techniques [55] have been applied, genetic algorithms [67–69, 73, 94–96], simulated annealing, or other heuristics [74, 97]. The efficiency of heuristic methods based on tabu search and steepest ascent are compared with limits obtained by MILPs [98]. Zuo and Pitts [99] investigated the influence of link cost ranges on optimization results but not in the context of USP routing. Riedl et al. increase the search space by considering non-additive link costs [71]. Xu et al. developed a new routing protocol based on link costs PEFT which enables optimal traffic engineering [72].

2.4.2 Tiebreakers and Unique Shortest Paths

In case of ECMP, traffic is only approximately evenly split over equal-cost paths towards a desired destination [57]. Thorup and Roughan [58] investigated this problem. To account for the load fluctuations, they added a multiplicative penalty of 20% more load on the links when a traffic aggregate is split over equal-cost paths and respected that in the objective function for routing optimization. This

already led to a reduction of path splits. They integrated this concept in the local search technique of [55] and compared ECMP and USP performance considering the failure-free case in the same networks as used in [55]. Lucraft et al. [65] also generated USP routing solutions based on the local search technique of [55] and used multiplicative and additive penalties in the objective function for equal-cost paths. In the algorithm presented here only the additive approach is adopted. Peterson et al. create symmetric USPs without any optimality goal using constraint programming [100]. The paths just needed to be able to carry a given traffic matrix. Zhang [101] proposes a new mathematical formulation of the USP problem which can be solved with constraint generation algorithms. In [102], different exact and heuristic solution methods based on *Integer Linear Program (ILP)* formulations were compared to obtain USP solutions for optimal IP routing under failure-free conditions and it was shown that the exact mathematical methods can solve only small problem instances. A simple heuristic based on [54] specialized for failure-free routing mostly led to better solutions than upper bounds received by the exact methods. Bley proposes a decomposition approach to find optimal USPs [103]. In a first step an optimal path routing is computed using integer programming techniques and in the second step link cost settings are determined that induce this routing. In [104] he considers the cost-optimal design of IP networks with USP routing. To the best of knowledge, none of the papers has tried to produce optimal USP solutions for resilient IP networks which is the objective addressed in Section 2.2.

2.4.3 Fast Reroute Mechanisms

A framework for IP-FRR [51] was recently published as RFC by the IETF routing working group. The NotVia mechanism was published in [34] and improvements have been proposed in [105]. As mentioned before, there are multiple other proposals for the implementation of IP-FRR, such as, e.g., LFAs [43], Remote LFAs [44], MRTs [45, 46], and MRC [47]. The authors of [50] give an extensive overview on MPLS and IP-FRR mechanisms. Further overviews can be found,

e.g., in [48, 49] and the references within. FRR concepts were first developed for MPLS technology and standardized in [35]. Extensions for point-to-multipoint were under discussion to protect multicast traffic [106, 107]. The ability of plain IP routing for sub-second reaction to failures was studied in [108, 109] as well as stability issues when performing such optimizations.

Different work address the analysis and optimization of particular FRR mechanisms, such as the failure coverage of LFAs [3, 42, 48, 110–118] and remote LFAs [119]. Further information on LFAs as well as different modifications to it can be found in the mentioned publications and references within. Recently, Menth et al. [49] compared the performance of NotVia and MRTs regarding path lengths and maximum relative link load. To the best of knowledge, the NetOpt tool including the extensions presented in this chapter is the first to address the optimization of the maximum relative link load regarding also the IP-FRR NotVia mechanism.

2.5 Lessons Learned

This chapter focused on the optimization of IP-based resilient routing. Routing layouts based on IP shortest paths do not only exist in common intra-domain routing protocols, such as OSPF or IS-IS, but can also be configured in MPLS networks. The advantage of such routing layouts compared to other layouts is that they can be automatically calculated and are robust against failures. When adding IP-FRR or MPLS-FRR mechanisms, furthermore, a fast reaction on outages can be included. In this chapter, two issues related to IP-based path layouts have been discussed: i) *Unique Shortest Path (USP)* layouts allowing to avoid unambiguous path layouts and ii) a comparison of IP-based and explicit MPLS path layouts concerning optimization potential and configuration effort.

When several equal-cost shortest paths exist between two nodes in IP networks, *Tie-Breakers (TBs)* decide which of the paths is chosen for *Single Shortest Path (SSP)* routing. However, the TBs are not clearly defined and possibly do not work in a deterministic way. It was shown that this can lead to unexpected

high link loads in the examined networks in spite of routing optimization. In the networks considered in this chapter, the unexpected increase of the maximum relative link load reached values up to over 180% higher than expected during the optimization. This values highly depend on the considered network and link cost settings and cannot be generalized. However, they indicate that TE is useless when relying on unknown TBs. To avoid problems with non-deterministic TBs, it was proposed to use only link costs that lead to USPs, i.e., only a single shortest path exists between a source and destination pair. It was shown that the fraction of (random) link cost settings that produce USP solutions depends on the maximum allowed link costs and on the set of protected failure scenarios.

One motivation for USP routing is traffic optimization. A heuristic algorithm was presented to find link cost settings that produce USP routing and to minimize the maximum relative link load in all protected failure scenarios for classic IP networks, NotVia IP-FRR, and MPLS networks with facility or one-to-one backup. The performance comparison showed that, compared to regular SSP, the additional path-uniqueness constraint of USP does not deteriorate the routing quality with respect to maximum relative link load and path length. It is a remarkable result that this holds even for the more constrained cases of resilient IP routing and FRR. ECMP often achieves the lowest relative link load but at the price of uncertain routing decisions for specific packets. These results hold in all considered Rocketfuel topologies. Hence, USP routing solutions for IP-based networks can be effectively found and optimized for various reroute techniques without impairing the performance of single path routing. FRR mechanisms in general lead to larger maximum relative link load values and to longer backup paths.

The second issue addressed in this chapter is a comparison of IP-based and explicit MPLS path layouts. MPLS and MPLS-FRR path layouts can realize any routing by setting up *Label Switched Paths (LSPs)*. If these LSPs are based on the underlying IP network, they can be configured automatically by sending initialization packets according to IP shortest paths. Explicit path layouts offer more freedom but need more configuration effort. In this chapter, three different path layouts were compared, namely *IP-based USP* (USP_{IP}) created by NetOpt as well

as *Explicit Single Path* (SP_{ex}) and *Explicit Multi Path* (MP_{ex}) obtained by MILPs presented in [1, 12]. The differences between the layouts in optimization potential and configuration effort were quantified for the same networks as used for the USP studies.

The results revealed that explicit single path layouts might lead to significantly lower maximum relative link loads than USP layouts. However, this strongly depends on the considered network. In the regarded networks, in average USP_{IP} layouts are about 6% worse than SP_{ex} . However, the results also showed, that if the MILP are not run to optimality but interrupted earlier due to too long computation time, values can be arbitrarily bad, and even worse than the heuristically optimized routing layouts based on IP shortest paths. When demands are split onto multiple paths, the maximum relative link load can be reduced in average by about 45%. However, this leads to an increase of the overall number of paths by up to 100%. In average over the considered networks, the maximum number of paths per node increases to up to 267% of the value for USP_{IP} . The evaluation shows that the decrease of the maximum relative link load during the routing optimization usually comes at costs of a high configuration effort. Depending on the particular needs and use case of a network, the operators have to choose which type of routing layout is most adequate.

3 Analysis of Optimization

Effectiveness and Extension to the Full Failure Cycle

This chapter focuses on the analysis of the effectiveness of link cost optimization and on the extension of the optimization to the full failure cycle including all phases of link state routing: failure-free phase, *Fast Reroute (FRR)* phase, re-converged phase during failures, as well as convergence between the different phases¹. Two particular research questions are addressed which will be briefly explained in the following: i) how good does a routing optimized only for a small set of failures perform in other failure scenarios and ii) what is the influence of the *Loop-Free Convergence (LFC)* phase on the maximum relative link load of IP-based routing and how can the LFC be included in the optimization process.

The optimization of resilient routing involves the computation of the routing for many different failure scenarios. To keep this process computationally feasible, however, only a limited set of the most probable failure scenarios can be examined. In this monograph, mostly the set consisting of the failure-free case as well as all single link failures is regarded. Considering this fact, the first issue addressed in this chapter is whether the optimization for that limited set of scenarios performs well also for other possible failures. To address this question, a framework for resilience analysis proposed by Menth et al. [39, 40] is used and the ResiLyzer tool implementing this framework is extended by further visualiza-

¹Further explanation of the full failure cycle is given in Section 3.2.

tions. With an example analysis of optimized and unoptimized routing with and without *IP Fast Reroute (IP-FRR)* in the Exodus network used in Chapter 2, it is shown that optimization considering only single link failures also significantly improves the performance for scenarios with multiple simultaneous failed components. Together with the example analysis, the used resilience analysis concept is briefly introduced and the functionality of the ResiLyzer tool is shown.

The second issue addressed in this chapter is the inclusion of the LFC phase in the routing optimization and the optimization of the full failure cycle including all phases of link state routing. As explained before in Chapter 2, IP routing adapts to the failure state and converges to a new routing layout automatically. As this update can take up to several seconds, FRR mechanisms provide precalculated backup paths so that each router can instantaneously shift the traffic to other paths when detecting a failure. The use of *Unique Shortest Path (USP)* routing layouts assures that the routing layouts are clearly defined and no ambiguous equal-cost paths exist. During the convergence phase between failure-free routing and routing in the failure state, micro-loops might appear temporarily, when routers update their *Forwarding Information Base (FIB)* in an unfavorable order and packets loop between routers with different new and old routing information. Special LFC mechanisms avoid these loops by defining certain constraints for the FIB update order of the routers. However, a problem of LFC mechanisms is that while these mechanisms are used, the relative load of certain links may significantly increase. Some links might even experience overload and subsequent packet loss. This behavior is investigated using *Not-via addresses (NotVia)* [34] as FRR mechanism and *Ordered FIB Updates (OFIB)* [41] as LFC mechanism. The investigations show that OFIB often temporarily increases the relative load on certain links in the network, possibly causing overload on these links. The impact of OFIB on the maximum relative link load is quantified for several well-known example topologies already used in Chapter 2. The observations show that the temporary increase of the relative link load can be significant and is a non-negligible issue. It is especially important to consider this for the optimization of resilient routing trying to reduce the maximum relative link load also during

certain failure cases. Therefore, the second part of this chapter addresses how to include the temporary load increase caused by OFIB in the optimization.

The OFIB concept does not provide a fixed update order but only provides certain constraints. The number of OFIB-conform update orders is exponential with regard to the number of routers in the network, so it is computationally not feasible to analyze all possible update orders. It is first tried to find the maximum relative link load during the OFIB phase by simulating a number of different valid update orders. Then, an algorithm is introduced that finds an order-independent upper bound to this maximum relative link load. The upper bound is tight, can be computed fast, and allows for the extension of the heuristic routing optimization using NetOpt presented in Chapter 2 to consider the OFIB process. The work presented in this chapter extends typical work in the area of routing optimization by providing a resilient IP routing optimization framework that takes into account not only the failure-free, FRR and failure case but also the LFC phase.

The first part of this chapter is mainly taken from [2, 13, 14]. The discussion on the full failure cycle and LFC using OFIB is mainly taken from [15]. The remainder of this chapter is as follows. Section 3.1 discusses the analysis of the optimization effectiveness of link cost optimization. First, an overview on resilience analysis is given. Then, the likelihood of overload for unoptimized conventional IP rerouting and for IP-FRR is compared and the impact of routing optimization also for IP-FRR is illustrated. In Section 3.2, the extension of the optimization to the full failure cycle including the LFC phases is discussed. First, failure handling in link state routing is briefly addressed, its different phases are summarized, and OFIB are explained. Then, the problem of temporary load increase during LFC is illustrated, analyzed and quantified. An upper bound algorithm is introduced to efficiently analyze OFIB and consider it during optimization. Finally, the results of routing optimization including different phases of the full failure cycle are presented and discussed. Section 3.3 gives an overview on related work. The chapter is concluded and the results are summarized in Section 3.4.

3.1 Effectiveness of Link Cost Optimization

In this section, the effectiveness of Link Cost Optimization is investigated. A resilience analysis framework introduced in [39,40] is used. First, this framework is briefly explained. Then, the ResiLyzer tool implementing the framework is presented. Finally, conventional IP routing and optimized routing are compared regarding both, normal IP rerouting in failure cases and IP-FRR.

3.1.1 Resilience Analysis

Link and router failures may lead to disconnection of nodes within a network and to rerouted traffic causing increased load on backup paths. Resilience analysis quantifies the disconnection probability of nodes due to failures and the potential overload caused by backup traffic or abnormal traffic demands.

Required inputs are the network topology, the routing and rerouting model, the link capacities, an availability model for network elements indicating failure probabilities as well as a model of the traffic matrix indicating the probability and the structure of abnormal traffic demands. Networking scenarios $z = (s, h)$ are defined consisting of a failure scenario $s \in \mathcal{X}$ and a traffic matrix h . Failure scenarios and traffic matrices are associated with probabilities $p(s)$ and $p(h)$. It is assumed independence, so that the probability of a networking scenario can be calculated by $p(z) = p(s) \cdot p(h)$.

The idea of the analysis framework is to investigate the disconnection of nodes and relative link loads for individual networking scenarios z . These results then contribute with a probability weight of $p(z)$ to the final result. Due to computational limitations, it is not possible to consider all possible failure scenarios and traffic matrices. Therefore, the analysis considers only networking scenarios with a probability of at least p_{min} and this set is denoted by $\mathcal{Z} = (\mathcal{X}, \mathcal{H})$. The final results of the analysis are probabilities for the disconnection of a given node pair due to failures and *Complementary Cumulative Distribution Functions (CCDFs)* of the relative load for each link in the network. Both, the disconnection proba-

bilities and the CCDF of the relative link load values, are conditional in the sense that they refer only to the set of investigated scenarios \mathcal{Z} . However, the resilience analysis framework allows to provide upper and lower bounds on the true probabilities and relative link load values. In the following, this aspect is omitted for the sake of simplicity. In this chapter, only network element failures are considered as source for increased traffic on links, and only a single standard traffic matrix without anomalies is used.

There are several possible applications of resilience analysis. Using this technique, operators can check if the network's current state is sufficient to allow additional clients, to sell better Service Level Agreements, or to deal with the traffic increase possibly arising in the next few months. If the current network state is not sufficient, the resilience analysis can help to decide where to add new links or routers. Furthermore, resilience analysis can be used to study the influence of a new routing or to investigate the effectiveness of routing optimization on potential overload as done in this chapter.

3.1.2 The ResiLyzer Tool

The ResiLyzer has been developed to implement the previously presented resilience analysis concept into a software tool. An analysis with the ResiLyzer normally consists of four steps. First, the necessary input data is provided by loading existing topology, traffic matrix, and link cost files or by creating new ones via the corresponding panels or menu bars. Second, the relevant networking scenarios including effective topologies and traffic matrices are configured. Third, the general analysis is invoked and the analytical results are computed. Fourth, the analytical results are interpreted by choosing one of the proposed comprehensive views or exporting the raw data for further analysis.

Program structure

The ResiLyzer is implemented as an Eclipse RCP application. All elements of the tool are modular which makes them easily extensible. Figure 3.1 shows an

overview on the program structure. Each module is displayed together with its main features that are currently implemented. The application core is formed by the Eclipse RCP application and the corresponding GUI. There are currently four input modules: modules for topologies, traffic matrices, link costs, and routing. The ResiLyzer has been equipped with a large collection of precalculated example scenarios including the Rocketfuel topologies [120] and a selection of random topologies created with the Waxman model [121]. These scenarios consist of the topologies, corresponding traffic matrices created with a simple gravity model [122], and link costs optimized with the optimization heuristic NetOpt presented in Chapter 2. The currently implemented routings of the ResiLyzer include *Equal-Cost MultiPath (ECMP)*, *Open Shortest Path First (OSPF)* as well as USP as addressed in Chapter 2. Additionally, the *NotVia* and *MPLS Fast Reroute (MPLS-FRR) facility* and *one-to-one backup* mechanisms have been implemented.

The calculation of the considered networking scenarios z including failure scenarios s and traffic surges h is realized by special modules. Failure scenarios can be created either probabilistic with a threshold p_{min} or by selection of failure types, e.g., all single link and node failures. The currently supported types of traffic scenarios are hot-spot scenarios and interdomain rerouting scenarios.

The interpretation and illustration of the analytical results are performed by the unavailability and the overload module. The tool offers different views and graphs to allow for a simple monitoring of fault-tolerance. The user can reach a certain view intuitively by selecting the corresponding element, e.g., links or failure scenarios. For instance, selecting a failure scenario shows the unavailability and load situation in the network in this failure scenario, selecting a link shows the relative link load of this link in all failure scenarios.

Graphical User Interface and Use Cases

Figures 3.2 and 3.3 show different screenshots of the ResiLyzer *Graphical User Interface (GUI)* illustrating some of the use cases of the ResiLyzer. The GUI

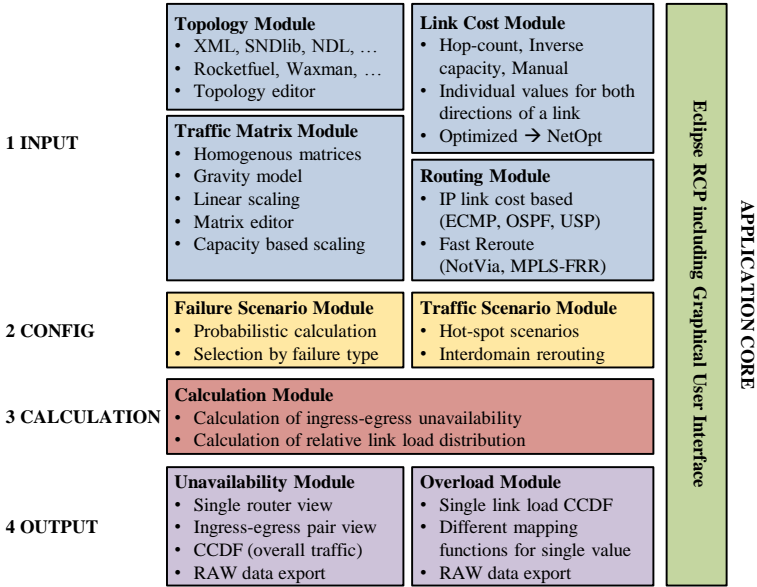


Figure 3.1: Program structure of the ResiLyzer.

allows the user to highly adapt one’s personal view on the tool, selecting the needed menus, panels, windows, etc. In Figure 3.2, it is shown how the relative link loads in the network in different scenarios can be analyzed. The left topology displayed in the screenshot represents the failure-free case, the right topology the outage of one of the nodes. In the lower part of the application window, a list of all links and their properties can be seen.

In Figure 3.3(a), an example for disconnectivity analysis is depicted. By adding additional links to the network, the probability of the disconnectivity between single nodes can be reduced. In the displayed example, the nodes are colored according to a traffic light color scheme indicating the risk that a particular

3 Analysis of Optimization Effectiveness and Extension to the Full Failure Cycle

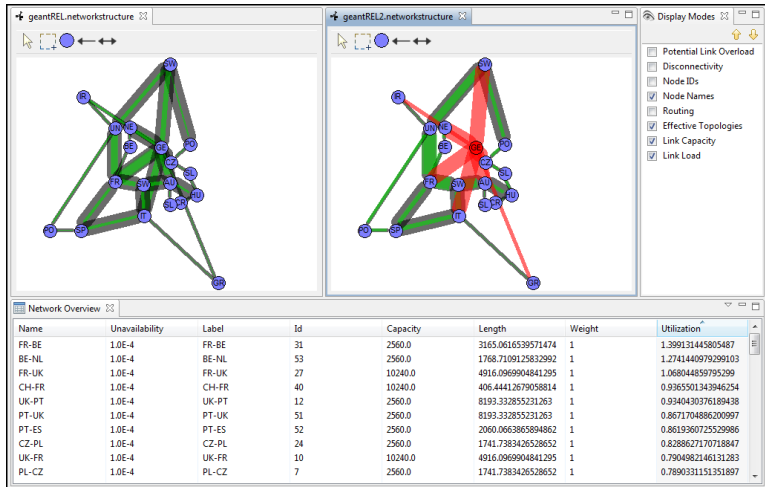
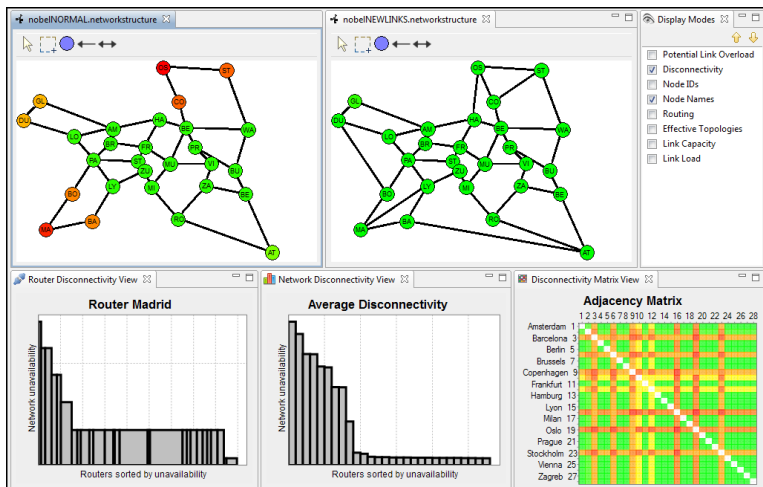


Figure 3.2: Screenshot of ResiLyzr GUI showing relative link load analysis.

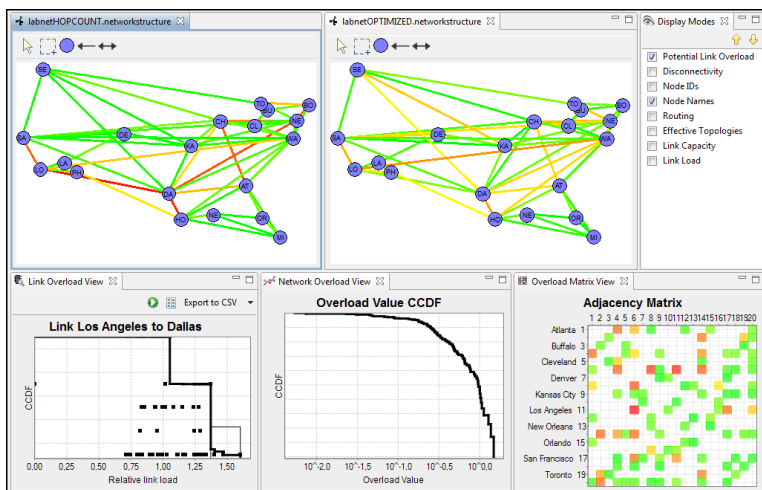
node is disconnected from the rest of the network. An unoptimized (left side) and an optimized (right side) network are compared.

The last use case displayed here, see Figure 3.3(b), is overload analysis. Similar to what was explained above, ResiLyzr also allows to color links in the network according to the probability that these links exceed certain relative link loads. In the displayed example of the Labnet network used before, unoptimized and optimized routing are compared regarding the overload probabilities of certain links. In the lower part additional views are shown that allow for a more precise analysis of particular effects. Overload analysis is the use case addressed in the following section, when regarding the effectiveness of link cost optimization. Further information on the ResiLyzr can also be found online [123].

3.1 Effectiveness of Link Cost Optimization



(a) Disconnection analysis.



(b) Overload analysis.

Figure 3.3: Further ResiLyzer example use cases.

3.1.3 Results

In the following, the effectiveness of routing optimization for IP routing and IP-FRR are studied using a concrete example scenario. Therefore, unoptimized *Hop-Count (HC)* routing is analyzed first and then it is compared to optimized USP routing. It is shown that even the link cost optimization with a small set of protected failure scenarios \mathcal{L} leads to routings that significantly improve the overall resilience of the network. In a second step, the difference between unoptimized and optimized routing using NotVia IP-FRR techniques is investigated.

Scenarios under Study

The study is conducted and the results are illustrated with the example of the Exodus network used in Chapter 2 and illustrated in Figure 3.4. The *Traffic Matrix (TM)* used for the analysis in this section has been created similar to the matrices used in Chapter 2. It is resembling real-world data according to the method proposed in [63] and enhanced in [64]. All links were expected to have identical capacity and in this case, the TM was scaled so that the worst relative load experienced by a link in case of single link failures and HC routing is 75%. However, similar to the link cost optimization case, relative link loads larger than 100% can be achieved in single node and multiple failure scenarios. An unavailability of 10^{-6} for all nodes is chosen. Each link is unavailable with the same probability of 10^{-4} . The set of investigated scenarios \mathcal{Z} has been calculated for $p_{min} = 10^{-15}$. This results in a number of $|\mathcal{Z}| = 51577$ considered scenarios, about a thousand times more, than the number of single link failures considered for the link cost optimization $|\mathcal{L}| = 51$. \mathcal{Z} consists of the failure patterns $\emptyset, L, N, LL, LN, NN, LLL, LLN$, where L denotes a single link and N a single node failure. A resilience analysis with the described settings reaches very high precision, while still being computationally feasible². The optimized USP

²Such a resilience analysis can be usually conducted in a time scale of a few minutes. On the other hand, the optimization of some of the link cost results obtained with NetOpt took several days even though only single link failures were considered during the optimization.

routing and NotVia solutions regarded in the following correspond to the best solutions for the Exodus network in the result database of all optimization results described in Chapter 2.

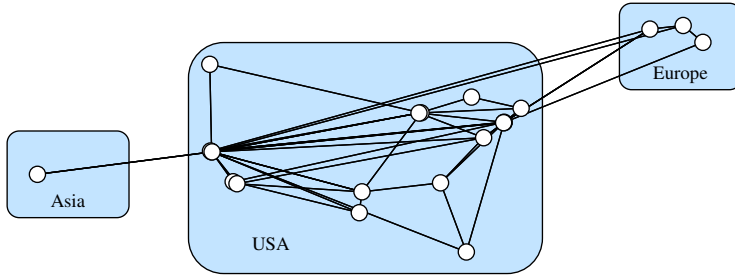


Figure 3.4: Exodus network, 22 nodes, 102 unidirectional/51 bidirectional links.

IP Routing and Rerouting Based on the HC Metric

In the following, the potential overload in a network is analyzed when HC routing is used. The relative load for the link from Palo Alto to Santa Clara is investigated because its potential overload is especially high in some failure scenarios. Figure 3.5 shows the CCDF of the relative link load $\rho(l)$ for this link. The CCDF illustration simplifies the observation of the potential overload for a single link.

The probability $P(\rho(l) > x)$ that a relative link load $\rho(l)$ exceeds a certain value x is directly displayed in the graph. In this case, e.g., the probability that relative link loads higher than 60% occur from Palo Alto to Santa Clara is about 0.06% $P(\rho(l) > 0.6) \approx 0.06\%$. This value is later referred to as $R_r^{0.6}$. On the other hand, in at least 99.999999% of all scenarios the relative link load is not larger than about 1.16, $P(\rho(l) \leq 1.16) > 99.999999\%$. This value is later referred to as $R_q^{0.99999999}$. In particular, this is true for all single and double link failures as well as single node failures.

3 Analysis of Optimization Effectiveness and Extension to the Full Failure Cycle

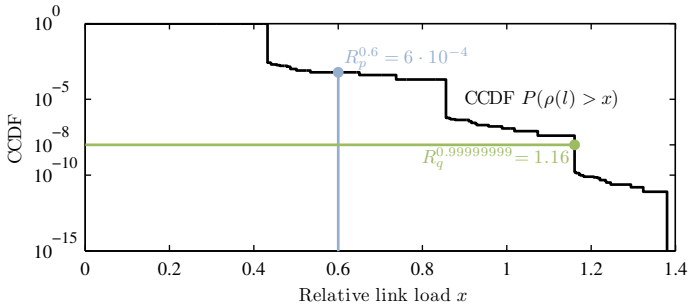


Figure 3.5: CCDF of the relative link load $\rho(l)$ for the link between Palo Alto and Santa Clara.

If CCDFs are used, a complete figure is necessary to visualize the probabilistic load condition on a link. Monitoring such information for all links in the network becomes more difficult with an increasing network size. Therefore, Menth et al. [39, 40] presented various mapping functions to aggregate the information of the per link CCDF into one value per link. Two of those functions are used in the following:

- Mapping function $R_p^x(l) = P(\rho(l) > x)$ is based on overload probabilities. It returns the probability with which the relative load $\rho(l)$ of link l exceeds the relative load value x . Figure 3.5 illustrates $R_r^{0.6}$.
- Mapping function $R_q^y(l) = \inf(x : P(\rho(l) \leq x) \geq y)$ is based on relative link load quantiles. This mapping function returns the smallest relative link load value x which is not exceeded by a fraction of at least y of all considered network scenarios. Figure 3.5 depicts $R_q^{0.99999999}$.

The mapping functions are used to convert the CCDF of each link to a single value. Then, those values are mapped to a color scale indicating the severeness of the potential overload.

In Figure 3.6, the geographical view of the Exodus network is colored according to the quantile based mapping function $R_q^{0.9999999}$ for unoptimized HC routing and optimized USP routing. The colors of the links can be converted to numerical relative load values using the color bar on the right side of the graph.

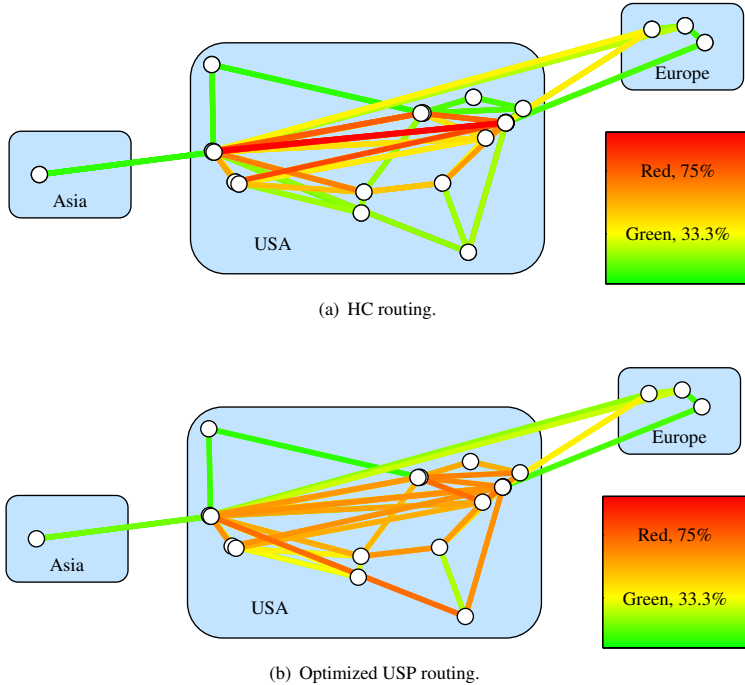


Figure 3.6: Exodus network colored according to the potential overload risk for different routings. The color of a link corresponds to the 99.999999% quantile of its CCDF of the relative link load. Darker colors indicate higher overload values.

This way of illustration allows for an easy geographical locating of critical network parts. It is however not the ideal illustration to add link or node related

information. Some nodes are so close to each other that they cannot be differentiated. Forward and backward directions of links cannot be distinguished, either. In the displayed example, for each pair of forward and backward links, only the color of the one of both links with the higher overload value is shown. No information about the color of the second link can be read from the graph. To address this problems, an adjacency matrix to represent the network topology as in Figure 3.7 is proposed. The cell of row i column j in the adjacency matrix corresponds to the link between nodes i and j .

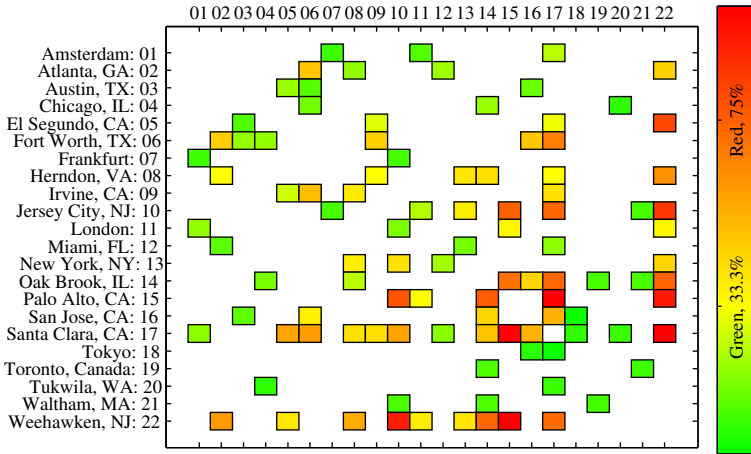
Figure 3.7 shows the adjacency matrix of the Exodus network colored according to the same color scheme as used in Figure 3.6. This illustration shows the potential overload of the whole network and the link with the risk of highest overload can be directly recognized. Again, the colors in the tiles can be converted to numerical relative load values using the color bar on the right side of the graph.

In both, Figures 3.6 and 3.7, it can be seen that for optimized routing, there are less links with a very high risk of overload (red colored links). On the other hand, the number of links with a slight risk of overload increases (orange colored links). This fact, that is analyzed in more detail in the following, indicates that the routing optimization distributes the traffic in the network to other links, leading to a lower maximum relative link load in the failure scenarios considered during the optimization and indirectly also to a lower risk of overload in more severe failure scenarios.

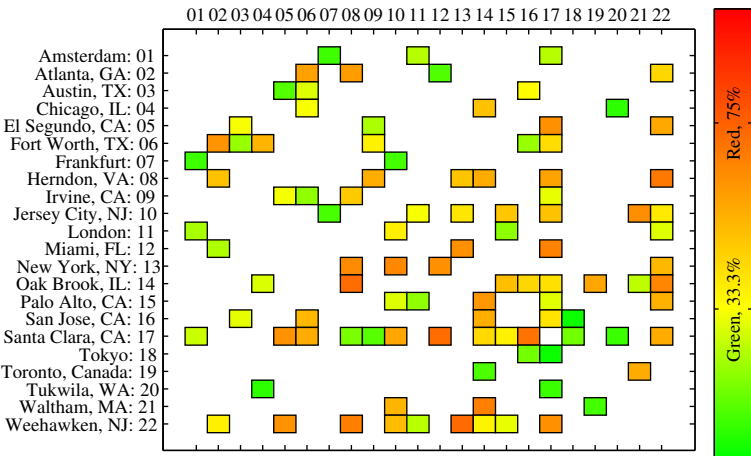
Optimized IP Routing and Rerouting

In the following, the impact of routing optimization on the potential overload is shown. Figure 3.8(a) shows the CCDF of the relative load on the link from Palo Alto to Santa Clara for HC routing and optimized USP routing. The curve of the optimized USP routing is at all values smaller than the one for HC routing. Thus, the routing optimization indeed reduces the overload risk on this particular link. As a consequence, all mapping functions yield smaller values for optimized USP routing than for HC routing. This findings hold only for this particular link the worst utilized for HC routing. The link, depicted in Figure 3.8(b), between Santa

3.1 Effectiveness of Link Cost Optimization



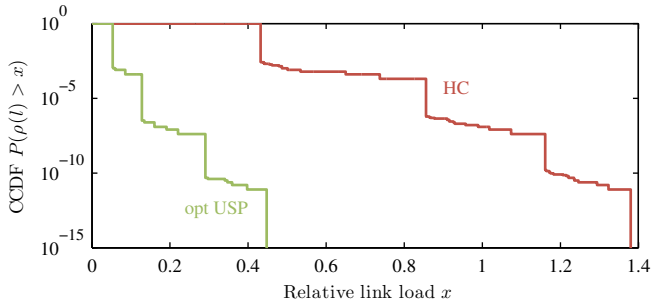
(a) HC routing.



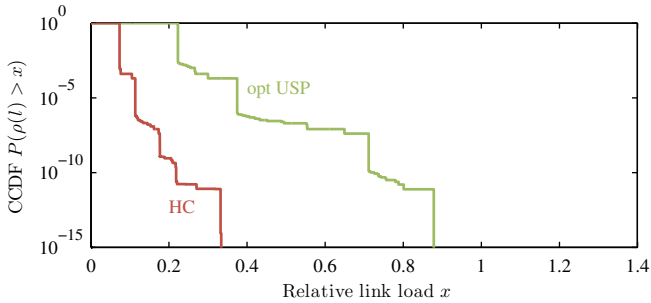
(b) Optimized USP routing.

Figure 3.7: Adjacency matrix of the Exodus network colored according to the same color scheme as used in Figure 3.6. The color of a link corresponds to the 99.99999% quantile of its CCDF of the relative link load.

3 Analysis of Optimization Effectiveness and Extension to the Full Failure Cycle



(a) Link from Palo Alto to Santa Clara.



(b) Link from Santa Clara to Miami.

Figure 3.8: *CCDF of the relative link load $\rho(l)$ for HC routing and optimized USP routing.*

Clara and Miami presents a special counter example. Here, the risk of overload is larger after routing optimization. An optimized path layout does not decrease the total amount of traffic in the network but just distributes it differently over the links. However, Figure 3.8(b) shows that the resulting load increase on some links does not cause any real problems because the relative link loads still remain relatively low.

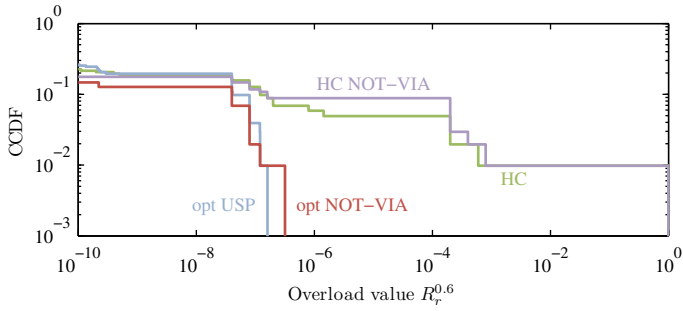
To visualize the impact of routing optimization on the potential overload, all links of the network need to be taken into account. Therefore, the overload values are calculated according to any mapping function R_p^x or R_q^y based on the CCDFs for all links. Then, the fraction of links is specified, whose potential overload exceeds a certain value. This leads to a CCDF of the overload values of the chosen function R_p^x or R_q^y .

Figure 3.9 shows CCDFs of overload values according to both mapping functions for HC routing and optimized USP routing. The lines corresponding to NotVia routing will be addressed in the following paragraph. Routing optimization redistributes the traffic in the network. On the one hand, this leads to a reduction of the worst overload values in the network. On the other hand, on some links with lower potential overload the values lightly increase. This effect is clearly visible in both graphs and also corresponds to what was addressed before regarding Figures 3.6 and 3.7. The result holds for both mapping functions. The elimination of the links with the highest risk of overload and the redistribution of the traffic to other links show that the link cost optimization on a small set of protected failure scenarios \mathcal{L} is very effective because it significantly improves the resilience calculated on a large set of scenarios \mathcal{Z} .

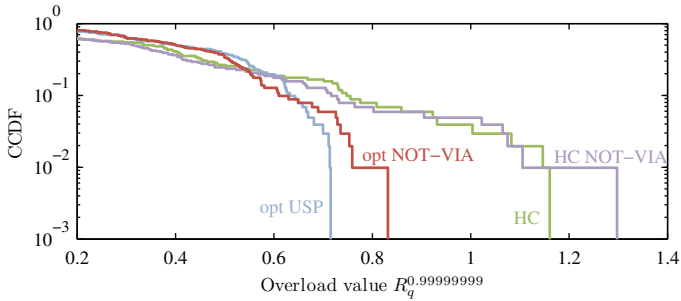
IP Fast Reroute Method Not-Via

Additionally to conventional IP routing, NotVia IP-FRR based on HC routing and based on optimized USP routing is investigated. Therefore, the overload values of the entire network are compared for HC routing and optimized USP routing to unoptimized and optimized NotVia IP-FRR. Figure 3.9 displays the overload values of NotVia IP-FRR. The potential overload in case of unoptimized NotVia IP-FRR is even higher than for conventional IP HC routing. Routing optimization significantly improves these values. Optimized NotVia IP-FRR reaches overload values of similar quality as optimized USP routing. This holds for both mapping functions $R_p^{0.6}$ and $R_q^{0.99999999}$. As expected from the results presented in Chapter 2, the overload values caused by NotVia IP-FRR are higher than for con-

3 Analysis of Optimization Effectiveness and Extension to the Full Failure Cycle



(a) CCDF over all links of the probability that a relative link load exceeds 60%.



Color bar of Figures 3.6 and 3.7. Green, 33% Red, 75%

(b) CCDF over all links of the 99.999999% quantile of the relative link load.

Figure 3.9: Comparison of the CCDFs of the potential overload for IP rerouting and NotVia IP-FRR.

ventional IP routing especially due to the increased load on backup paths and the longer average path lengths due to local repair. However, routing optimization can help to reduce the risk of overload to a secure level also for NotVia IP-FRR.

3.2 Extension of the Optimization to the Full Failure Cycle

In Chapter 2 and the previous section, it was shown that routing optimization can efficiently reduce the maximum relative link load and the risk of overload for conventional IP routing and IP-FRR. According to the last section, this holds even when considering only a relatively small set of failure scenarios, such as all single link failures $\mathcal{X} = \mathcal{L}$. Furthermore, in Chapter 2, it was shown that IP-based routing involves a significantly lower configuration effort than explicit path layouts. Altogether, this leads to IP-based routing optimization being a promising candidate for *Traffic Engineering (TE)* in core communication networks.

In this section, a particular issue concerning IP routing is discussed: *Loop-Free Convergence (LFC)* and the optimization of the full failure cycle. First, the failure handling in link state routing is briefly repeated and the full failure cycle is introduced. Then, the problem of a temporarily load increase during the LFC phase using OFIB is quantified. The maximum relative link load during the OFIB phase is analyzed and included in the heuristic optimization with the NetOpt tool. This finally leads to a possibility to optimize all stages of the full failure cycle. According to the results of Chapter 2, ambiguous path layouts need to be avoided to assure the quality of the conducted optimizations. Therefore, in the following, only routing layouts fulfilling the *Unique Shortest Path (USP)* constraints are considered.

3.2.1 Failure Handling in Link State Routing and the Full Failure Cycle

In Section 2.1, the fundamentals of IP routing and FRR have already been explained. In typical intra-domain networks the routing is calculated autonomously by the routers by exchanging information on topology and link costs. Furthermore, IP routing is robust against failures and automatically converges to a new routing layout. As the reconvergence process can take several seconds, during

which traffic is lost, FRR mechanisms offer a fast reaction in case of outages. In the following, the FRR mechanism NotVia is used as it provides 100% failure protection coverage for all single link and router failures. Other FRR mechanisms fulfilling this criteria could be used as well. The following discussion is focusing on the convergence phases between failure-free and failure state. Therefore, the entire failure handling procedure is summarized and the full failure cycle is presented.

Phases of the Failure Handling Procedure As originally presented by Martin [56], the entire failure handling procedure in link state routing protocols like the *Intermediate System to Intermediate System Protocol (IS-IS)* or *Open Shortest Path First (OSPF)* can be divided in five different phases. These phases are illustrated as a failure cycle, see Figure 3.10.

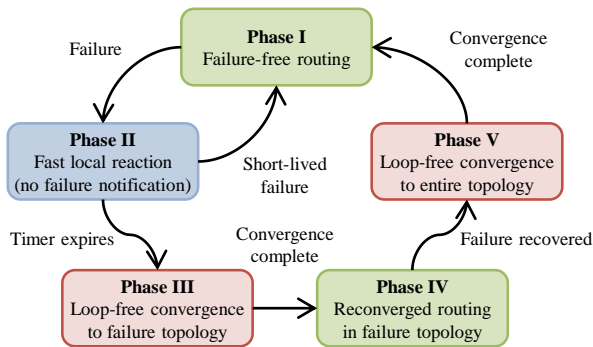


Figure 3.10: Phases of a link state routing failure handling procedure.

The normal operation state of any network is the *failure-free routing* (Phase I). All links and routers are operating normally and traffic is routed according to least-cost paths.

After a failure, the network enters the *fast local reaction phase* (Phase II). Most of the traffic is still routed according to least-cost paths. The traffic affected by

the failed components is sent on backup paths using a FRR technique. To avoid unnecessary network-wide reconvergence during short-term failures, the failure information is not broadcast in the entire network.

The use of FRR techniques increases the load on backup paths and leads to longer average path lengths. Furthermore, most FRR mechanisms are only designed to handle a single failure and subsequent failures would cause packet loss. Thus, after a preconfigured timer expires, a failure notification is distributed in the network and the network enters the IP reconvergence phase. This process (Phase III) can take several seconds. During this time, micro-loops might appear, where packets loop in between routers with different views of the network. When, for example, router *A* in Figure 3.11 updates its forwarding table after the failure of link $A \leftrightarrow B$, it uses *C* as next-hop towards *B*. But if router *C* has not yet updated, it sends these packets back to node *A* and a micro-loop is created.

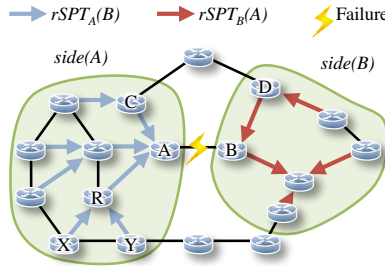


Figure 3.11: Reverse Shortest Path Tree (rSPT) towards a single link failure.

The existence of micro-loops is especially annoying when FRR mechanisms are used since in this case, initially almost no packets are lost. An uncontrolled reconvergence would however render the FRR detour useless, when the failure detecting router updates its FIB and sends packets to the new next hop. The subsequently occurring micro-loops could, in addition, lead to overload on other links and possibly impair otherwise unaffected traffic. Therefore, LFC has to be guaranteed. While the routers reconverge loop-free to the new routing, FRR mecha-

nisms continue to assure that no traffic is lost. In the reconverged routing phase (Phase IV), all traffic is routed according to least-cost paths in the failure topology. When the failure can be repaired, the routing converges back to the original failure-free state, again using a LFC mechanism (Phase V).

The discussion presented in the following particularly addresses the convergence phases (Phases III and V) and extends the heuristic optimizer NetOpt to include all stages of the failure cycle. The work of Martin [56] introduced the failure cycle but did not address the convergence phases in any more detail. In his work, particular issues related to failure coverage with a combination of *Loop-Free Alternates (LFAs)* and NotVia were regarded what is related to phases I, II and IV of the failure cycle.

Ordered FIB Updates

In the following, *Ordered FIB Updates (OFIB)* [41, 124] are used as LFC mechanism. OFIB assure LFC by imposing certain rules on the update order of the routers in the network. The general idea of OFIB is explained using the example in Figure 3.11 where a link fails and then reappears again. These events are called link-down and link-up event, respectively.

Terminology First, the OFIB terminology is explained. A failure of the bidirectional link $A \leftrightarrow B$ can be regarded as two unidirectional failures of links $A \rightarrow B$ and $B \rightarrow A$.

The *reverse shortest path tree* $rSPT(B)$ of router B is formed by the shortest paths of all routers towards the destination B . In Figure 3.11, only $rSPT_A(B)$ is shown, the *reverse shortest path tree regarding a link* $A \rightarrow B$. It is the subtree of $rSPT(B)$ that is attached to the router A . Thus, it is formed by all routers whose shortest paths to B include the link $A \rightarrow B$. The $rSPT_B(A)$ is constructed likewise with all routers whose shortest paths to node A include $B \rightarrow A$. Each shortest path can contain at most one direction of a link $A \leftrightarrow B$, so $rSPT_A(B)$ and $rSPT_B(A)$ are disjoint.

With regard to the failure of a link $A \leftrightarrow B$, a network can logically be split into two sides, in the displayed example in sides $side(A)$ and $side(B)$ of the link failure $A \leftrightarrow B$. All routers that form $rSPT_A(B)$ and the links connecting them are located on $side(A)$ of the failure, while all routers of $rSPT_B(A)$ are located on $side(B)$. Routers that do not use the link $A \leftrightarrow B$ are not assigned to any of both sides.

Link-Down Event $A \leftrightarrow B$ After a link-down event of link $A \leftrightarrow B$, micro-loops can appear only if an already updated router sends packets to a router that has not updated yet. To assure LFC on $side(A)$, a router R has to postpone its update until all other routers that send traffic via R and $A \rightarrow B$ have updated their FIBs first. Hence, the updates are conducted starting from the leaves of $rSPT_A(B)$, so that the routers farthest from the failure update first, the ones next to the failure update last. This prevents micro-loops during the reconvergence process [124].

Link-Up Event $A \leftrightarrow B$ Link-up events are handled likewise. Similarly to the link-down event, the reverse shortest path trees $rSPT_A(B)$ and $rSPT_B(A)$ are considered³. In this case, the updates in the rSPTs are conducted starting from the roots. Router R on $rSPT_A(B)$ delays its FIB update until all predecessors on $rSPT_A(B)$, i.e., all routers that R uses to transmit traffic via link $A \rightarrow B$, have updated their FIBs. Again, LFC is assured.

Update Order OFIB is based on certain update order constraints, which can be achieved with two mechanisms [41].

The first technique is based on timers. Each router calculates its so-called rank in the rSPT and, depending on that, a certain waiting time before starting the update. Figure 3.12 depicts the ranks for all routers of an arbitrary $rSPT_B(A)$. The two numbers assigned to each of the routers indicate the rank for a link-down and a link-up event. In case of a link-down event, the leaves of the rSPT

³These are the same rSPTs as before where link $A \leftrightarrow B$ is working.

(*F*, *G*, *D*, *I*, and *J*) are the first to update, and therefore, have rank 1. After the configured maximum update time, all routers with the next rank (*C* and *H*) start their update. This process is continued until all routers including the root of the rSPT have updated. The update order in case of a link-up event works vice versa. The first router to update is the root of the rSPT, router *B*. It is followed by the routers of the next rank (*C*, *D*, and *E*), and so forth. To make sure that all update order constraints are fulfilled even if some router's update takes longer, the waiting times for the timer-based update have to be chosen sufficiently large.

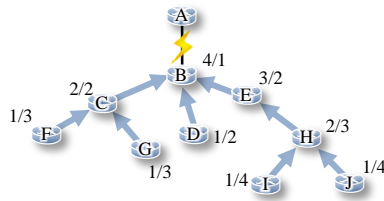


Figure 3.12: *Timer-based OFIB*. The two numbers assigned to each router indicate the update rank for a link-down and link-up event subsequently.

As the timer technique is rather slow, a second, message-based technique is proposed in [41]. In this case, each router *R* has a waiting list with other routers that still have to update before *R* and a notification list with routers that are waiting for *R*'s update. As soon as *R*'s waiting list is empty it updates and then, notifies all routers in the notification list. For example router *C* in Figure 3.12 waits for the updates of *F* and *G*, then updates, and finally, notifies *B* about its update. Using this technique, all routers can directly update as soon as the constraints are fulfilled, which significantly accelerates the OFIB process.

3.2.2 Temporary Load Increase Caused by OFIB Orders

OFIB solves the problem of micro-loops. However, during the LFC phase the step-wise updates can lead to a temporary relative load increase on certain links

in the network. In the following, two different types of a temporary increase of the relative link load are illustrated.

Looking at One Side of the Failure The first type of a temporary increase of the relative link load appears on a single side of the failure, e.g., $side(A)$ of link $A \leftrightarrow B$. It is caused by two or more nodes that can update independently of each other because they are in different subtrees of $rSPT_A(B)$, e.g., routers X and Y in Figure 3.13.

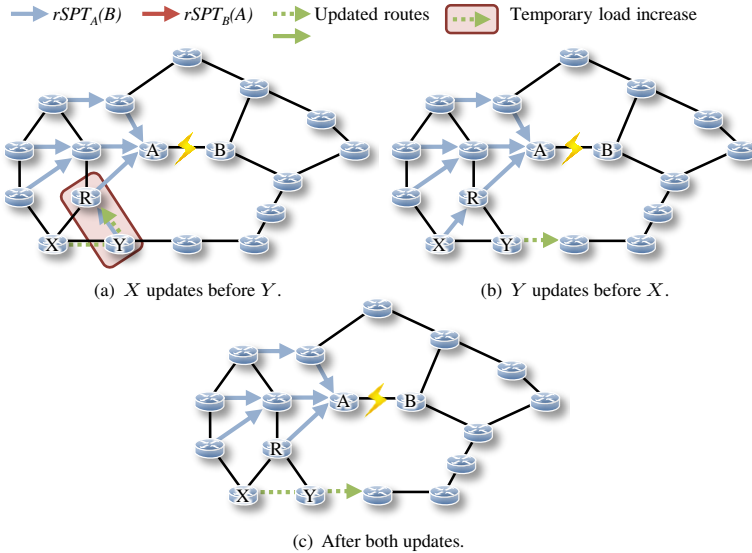


Figure 3.13: Example for a temporary increase of the relative link load on a single network side: When router X updates before router Y , see Figure 3.13(a), the network experiences a temporary increase of the relative link load on link $Y \rightarrow R$ that disappears after the update of Y , see Figure 3.13(c). This situation is avoided when Y updates first, see Figure 3.13(b).

Assuming that synchronous updates are technically not feasible, there are two possible update orders for X and Y that comply with the OFIB rules. Both orders are illustrated in Figure 3.13 and will be discussed in the following.

If X updates its FIB first, see Figure 3.13(a), Y sends the traffic originated in X via the old routing tree until it updates its own FIB. The network experiences a temporary increase of the relative link load on link $Y \rightarrow R$ that disappears after the update of Y , see Figure 3.13(c). This situation is avoided when Y updates first, followed by X , see Figure 3.13(b).

The example shows that OFIB may lead to unpredictable relative link load increase when independent routers incidentally update in a correct but disadvantageous order.

Interference between Different Failure Sides The second type of a temporary increase of the relative link load appears by an interference of different sides of a failure.

The update of a router on $side(A)$ in Figure 3.14 might cause a temporary increase of the relative link load on links of $side(B)$. Routers C and D can update independently as they are part of different rSPTs $rSPT_A(B)$ and $rSPT_B(A)$, respectively. Again, there are two possible update orders. The effects of both are displayed in Figure 3.14. In both cases the network experiences a temporary increase of the relative link load on a link on the other side of the updating router, i.e., an update on $side(A)$ influences a link on $side(B)$ and vice versa⁴.

If C updates first as in Figure 3.14(a), additional packets are routed over the link $D \rightarrow B$. Router D has not updated yet and also sends packets over the same link. After the update of D , the relative link load of link $D \rightarrow B$ decreases again because the new path from D to A does not include router B anymore, as depicted in Figure 3.14(c). If, on the other hand, D updates first as in Figure 3.14(b), the same temporary increase of the relative link load appears on link $C \rightarrow A$.

⁴There is additional load on some links caused by the FRR detour that is not shown in the figure.

3.2 Extension of the Optimization to the Full Failure Cycle

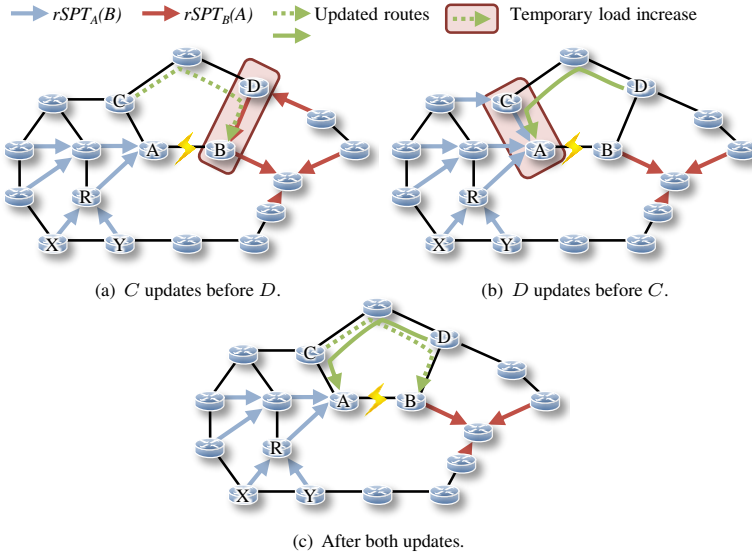


Figure 3.14: Example for interference between different failure sides: When routers of $rSPT_A(B)$ update before routers of $rSPT_B(A)$, a temporary increase of the relative link load appears on side(B) and vice versa. Changing the update order might not avoid this problem completely but shift the temporary increase of the relative link load to different links.

The example shows that updating routers on one side of the failure may cause a temporary increase of the relative link load on the other side. Not even regarding the necessary signaling overhead, it is hard or even impossible to automatically determine which side of the failure should start updating first.

3.2.3 Analysis of the Maximum Relative Link Load during the OFIB Phase

The previous explanations showed that OFIB can lead to a temporary increase of the relative link load. In the following, the topologies presented before in Chap-

ter 2 are used to quantify the impact of the temporary increase of the relative link load during the LFC phase on the maximum relative link load, i.e., the relative load of the most loaded link in the network. First, it is explained how the maximum relative link load during OFIB is calculated considering different possible update orders. Afterwards, numerical results are presented. To ease the analysis and handling of the temporary increase of the relative link load, an update-order-independent algorithm is presented that gives tight upper bounds to the maximum relative link load before, after, and during the LFC phase.

Calculating the Maximum Relative Link Load considering Different Update Orders

During the OFIB phase, many subsequent update steps are performed. To obtain the maximum relative link load including any temporary increase of the relative link load during OFIB, the relative link loads have to be calculated after each OFIB update step. The OFIB concept does not provide a fixed update order but only constraints that any possible update order has to fulfill. This is illustrated using Figure 3.15.

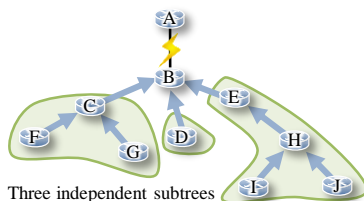


Figure 3.15: *Different subtrees of a rSPT. Each of these subtrees can conduct the OFIB updates independently of the others.*

The $rSPT_B(A)$ can be divided into three subtrees (C, F, G) , (D) , and (E, H, I, J) . Each of these subtrees can conduct the OFIB updates independently of the others. The root B has to wait until all subtrees have finished before starting

its update. Likewise, each node in a subtree has to wait until its children in the tree have updated. The duration of each update depends on many factors and is, in general, unknown. This leads to many possible OFIB orders. Similar to the different subtrees of router B , also the two sides of the failure, $rSPT_B(A)$ and $rSPT_A(B)$, can update independently, which further increases the number of update orders.

The number of OFIB-conform update orders grows exponentially with regard to the number of routers in the network. Thus, it is computationally not feasible to analyze all possible update orders and to obtain the worst case maximum relative link load. To estimate the maximum relative link load during the OFIB phase, following simulation procedure is chosen that can be conducted at acceptable computation effort: a set of 1000 different OFIB-conform update orders is created by varying individual update durations. These orders are evaluate step by step and the highest occurring maximum relative link load value is chosen as result of the simulation. The technique does not lead to the actual theoretical maximum relative link load. However, the worst value discovered for any of the 1000 random update orders represents a lower bound for this theoretical maximum value. In other words, if for at least one of the 1000 random update orders a temporary load increase during the OFIB phase is discovered for a given network and link cost setting, this illustrates that the problem of temporary load increase exists and gives a lower bound to the possible extent of the problem.

Numerical Results

The best link cost settings considering the maximum relative link load during the failure-free case \emptyset , all single link failures \mathcal{L} , as well as the NotVia IP-FRR case have been already used for evaluations in Chapter 2. Here, the same networks and link cost settings are evaluated using 1000 random update orders to estimate the maximum relative link loads $\rho_{\mathcal{L}}^{\max}$ also during the OFIB phase. To simplify the notation, following terminology is used: $\rho_{\mathcal{L},\text{FRR}}^{\max}$ represents the maximum relative link load during failure-free state, reconverged failure state, and

3 Analysis of Optimization Effectiveness and Extension to the Full Failure Cycle

NotVia state (Phases I,II,IV of the failure cycle). It does not include the relative link load during the OFIB phases (Phase III and V). $\rho_{\mathcal{L},\text{OFIB}}^{\max}$ additionally considers these phases and represents the maximum relative link load ever appearing during any phase of the full failure cycle. Figure 3.16 shows the $\rho_{\mathcal{L},\text{FRR}}^{\max}$ values compared to the highest simulated $\rho_{\mathcal{L},\text{OFIB}}^{\max}$ values in all networks under study. The effect of a possible temporary increase of the relative link load during OFIB on the maximum relative link load is different depending on the network and the link cost setting. In some of the networks, OFIB leads to no or little additional maximum relative link load. However, in AB, EX, and GE the increase is quite significant.

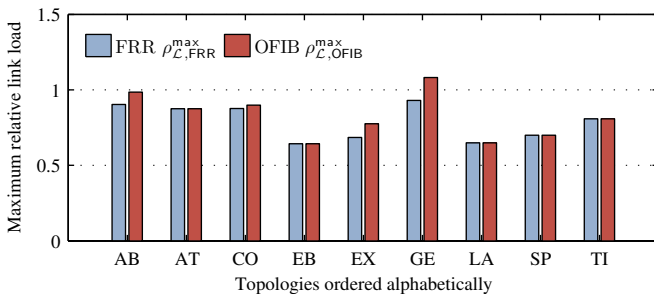


Figure 3.16: Maximum relative link loads during FRR and OFIB phase.

The analysis shows that OFIB can have a large impact on the maximum relative link load and therefore should be considered during the link cost optimization process.

Algorithm to Obtain a Tight Upper Bound

In the previous analysis, it was proposed to analyze the maximum relative link load during OFIB with a simulation of several random update orders. This has

two major drawbacks. First, the computation effort is large as several OFIB orders are regarded and need to be evaluated. Second, to effectively consider the OFIB maximum relative link load during routing optimization and to provide guarantees for the optimization results, the worst OFIB maximum relative link load has to be considered. However, there is no guarantee that the calculated values are even close to the worst OFIB maximum relative link load because only a small random subset of possible orders is analyzed⁵.

To provide a computationally fast calculation of the OFIB maximum relative link load that guarantees the quality of the optimization, an algorithm is proposed that provides update-order-independent upper bounds for the worst OFIB maximum relative link load. In the following, first, the algorithm is explained and then it is shown that the provided upper bounds are tight. Finally, the computational effort of the proposed algorithm is briefly discussed.

Algorithm description The basic concept of the algorithm is quite simple: during the FRR and LFC phase, the routing changes and traffic flows can be routed on different links. An upper bound to the maximum relative load of a link can be obtained by summing up the size of every flow that could be routed over this link in any possible network phase: the failure-free and reconverged case, the FRR case, as well as every possible OFIB update order.

Summing up all relative loads caused by all flows to a single link provides an upper bound but not necessarily a value that can really occur during a particular update order. Still, at the end of this section, it will be shown that the upper bounds are very tight for all considered networks.

To obtain all flows ever contributing to the relative load of a certain link, the relative link loads caused by each flow are calculated separately. Repetition of this procedure for all flows that are affected by the failure leads to an upper bound of the total maximum relative link load. Figure 3.17 illustrates the algorithm to calculate the upper bound on the network discussed before. The flow from X to

⁵This problem is of a similar nature as the problem of optimization with unknown Tie-Breakers discussed in Chapter 2.

3 Analysis of Optimization Effectiveness and Extension to the Full Failure Cycle

Z is considered, in case of a failure of link $A \leftrightarrow B$. As mentioned before, to avoid ambiguous path layouts, here, only Single Shortest Path routing based on link costs fulfilling the Unique Shortest Path constraints is considered. The figure shows a part of the original rSPT to Z , $rSPT(Z)$, in the failure-free case, and the new rSPT to Z in the failure-case $rSPT^*(Z)$, as well as the NotVia backup path from the *Point of Local Repair* (PLR) A to the Next-Next-Hop $NNHOP$.

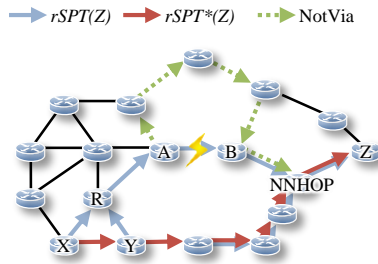


Figure 3.17: A single flow spread over all links it could use in any routing phase.

The algorithm starts at the source of the considered flow, router X . Two cases have to be considered: X has already updated the FIB or it has not update the FIB yet. In the former case, the flow X to Z is routed on $rSPT(Z)$. All routers on the path to the failure A have not yet updated to fulfill the OFIB constraints since they must wait for the update of X first. The algorithm adds the demanded traffic rate of the regarded flow to every link's relative load on the old path towards the failure (from X to A), on every link of the NotVia tunnel (from A to $NNHOP$), and on the subsequent links towards the destination (from $NNHOP$ to Z). In the second case, if X has updated its FIB already, the flow is sent to Y . Therefore, the algorithm adds the flow's traffic rate to link $X \rightarrow Y$.

At router Y , the same decisions are repeated again. If Y has not updated yet, the flow is sent on $rSPT(Z)$ via $Y \rightarrow R$. The algorithm has already added the flow to the relative link loads of the subsequent links before and therefore

does not add them again. If Y has already updated, the flow follows the updated routing on $rSPT^*(Z)$ to the next router where the described decisions are again repeated. As soon as $rSPT(Z)$ and $rSPT^*(Z)$ merge, the algorithm can be terminated as all possible links the flow might be routed on have been considered.

This procedure ensures that the traffic rate of the flow from X to Z is added to the relative link loads of all links that can ever transport it during the OFIB phase, regardless of the specific update order.

Tightness of the Upper Bound As mentioned before, the algorithm provides only an upper bound to the OFIB maximum relative link load. In the following, it is shown that this upper bound is mostly tight.

As described in Chapter 2, in course of different extensions to the heuristic optimizer, a database of all optimization results and corresponding link cost vectors for all considered networks was created. The results in this database are used for the evaluation. The upper bound algorithm is applied to all (more than 450.000) USP link cost settings in the database for the presented networks. The same link cost settings are evaluated using the previously described simulation of 1000 different OFIB-conform update orders and the worst found results are taken. Then, the relative difference between the obtained upper bound value and the worst OFIB value obtained by the simulated OFIB orders is calculated. Figure 3.18 shows the CCDF of this relative difference values over all evaluated link cost settings. A relative difference of, for example 30% indicates that the obtained upper bound value is 130% of the worst found OFIB value, i.e., 30% worse than this value.

For about 90% of the investigated link cost settings, the highest simulated OFIB relative link load is as high as the upper bound. Thus, for most link cost settings, the maximum relative link load of the upper bound algorithm represents a real value that can actually occur in the network. In more than 99% of the cases, the relative difference between the upper bound and the worst found OFIB value is less than 10%. Even in the cases where these values differ, the upper bound might still represent a realistic relative link load, because the simulation of OFIB

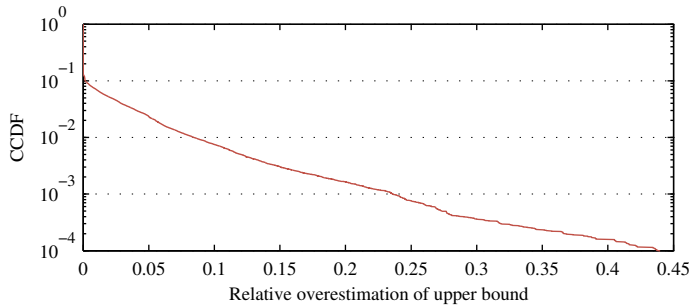


Figure 3.18: *Relative overestimation of the upper bound compared to the worst found OFIB value.*

uses only a limited number of update orders, and the actual worst case might not have been evaluated.

The evaluations show that the algorithm provides a good upper bound which permits the use for routing optimization.

Computational effort to calculate the Upper Bound The additional effort caused by computing the upper bound is almost neglectable. First, during the optimization process, the heuristic rejects solutions that are far from the current best value already in an earlier state. The OFIB upper bound is only computed for the few link cost settings that lead to NotVia maximum relative link loads that are equally good or better than the current best value found so far. Second, for the evaluation of the upper bound, no additional of the procedures involving the most computation time have to be run, such as the calculation of Dijkstra shortest path trees. The required original $rSPT$ and the new $rSPT^*$ are already computed when considering the failure-free routing and the routing in the reconverged state. The upper bound calculation consists only of placing the flows onto all possible links and on recalculating the maximum relative link load.

3.2.4 Optimization of the Maximum Relative Link Load during the OFIB Phase

The previous analysis has shown that the maximum relative link load can increase significantly during the LFC phase. In the following, it is shown that this effect can be minimized by link cost optimization.

The heuristic USP link cost optimizer is extended by implementing and integrating the upper bound algorithm presented for OFIB. This allows to consider the LFC phases during the optimization process. The link cost settings are optimized as before to reach USP layouts and to minimize $\rho_{\mathcal{L},\text{FRR}}^{\max}$. In addition, the OFIB relative link load, $\rho_{\mathcal{L},\text{OFIB}}^{\max}$, is optimized as a secondary goal. This way, it is expected to find link cost settings of equal maximum relative link load in the previously analyzed scenarios and, in addition, reduce the temporary increase of the relative link load during OFIB.

The efficiency of the algorithm is analyzed based on the database of all optimization results as described before. The best USP link cost settings in the database according to $\rho_{\mathcal{L},\text{FRR}}^{\max}$ and the best according to $\rho_{\mathcal{L},\text{OFIB}}^{\max}$ are compared. The former ones correspond to the ones regarded in the previous evaluations in Chapter 2 and Section 3.1.

Figure 3.19 uses an illustration similar to Figure 3.16. The first bar of each network shows the best maximum relative link load when link costs are optimized for $\rho_{\mathcal{L},\text{FRR}}^{\max}$ but not for $\rho_{\mathcal{L},\text{OFIB}}^{\max}$. These values are identical to the ones in the first bar of Figure 3.16. The second bar of each network shows for all link cost settings leading to the value $\rho_{\mathcal{L},\text{FRR}}^{\max}$ in the first bar, the worst $\rho_{\mathcal{L},\text{OFIB}}^{\max}$ value found in the database for these link cost settings using the upper bound algorithm (OFIB un-optimized). These bars represent upper bounds and are thus larger or equal to the worst found OFIB value bar in Figure 3.16. The third bar shows for each network the best $\rho_{\mathcal{L},\text{OFIB}}^{\max}$ value found in the database, i.e., the lowest found upper bound.

A comparison of the second and third bar indicates that there are link cost settings leading to the same $\rho_{\mathcal{L},\text{FRR}}^{\max}$ values but to worse $\rho_{\mathcal{L},\text{OFIB}}^{\max}$ values if OFIB is

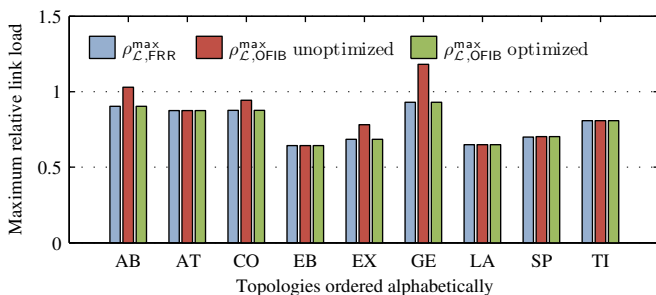


Figure 3.19: Comparison of the maximum relative link loads during FRR phase and OFIB phase with and without optimization.

not considered during the optimization. For all networks under study the first and third bar are equal. This shows that for all considered topologies, the extended NetOpt heuristic is able to avoid the maximum relative link load increase during the OFIB phase without impairing the maximum relative link load during the NotVia IP-FRR phase.

3.2.5 Optimization of the Full Failure Cycle

The previously presented discussion showed, with the example of OFIB, that the LFC phases can be efficiently included in the link cost optimization. In the following, the routing optimization including different phases of the full failure cycle is summarized. Therefore, for each considered network, link cost settings with different degree of optimization are compared.

Following degrees of optimization are looked at:

1. Unoptimized *Single Shortest Path (SSP)* HC routing. As discussed in Chapter 2, the use of ECMP is not possible for FRR backup paths. Therefore, SSP is regarded. As known from Chapter 2, the presented HC val-

3.2 Extension of the Optimization to the Full Failure Cycle

ues present only the maximum relative link load for one possible *Tie-Breaker (TB)* and can differ if other TBs are used.

2. Optimization only regarding the failure-free case \emptyset (Phase I of the full failure cycle): ρ_{\emptyset}^{\max}
3. Optimization additionally for reconverged IP routing in all single link failures \mathcal{L} (Phases I and IV): $\rho_{\mathcal{L}}^{\max}$
4. Optimization additionally for NotVia FRR (Phases I, II, and IV): $\rho_{\mathcal{L},FRR}^{\max}$
5. Optimization additionally regarding OFIB (Phases I to V): $\rho_{\mathcal{L},OFIB}^{\max}$

To compare the different cases, link cost settings from the result database described before are investigated. Following settings are chosen from the database to represent the different optimization degrees as good as possible:

1. HC link cost settings for all networks, in the following abbreviated as H.
2. The best USP link cost setting for each network according to ρ_{\emptyset}^{\max} , abbreviated as \emptyset .
3. The best USP link cost setting for each network according to $\rho_{\mathcal{L}}^{\max}$, abbreviated as \mathcal{L} . In case several link cost settings leading to the same best value exist, the one with the lowest value of ρ_{\emptyset}^{\max} is chosen.
4. The best USP link cost setting for each network according to $\rho_{\mathcal{L},FRR}^{\max}$, abbreviated as F. In case several link cost settings leading to the same best value exist, the one with the lowest values of $\rho_{\mathcal{L}}^{\max}$ and ρ_{\emptyset}^{\max} is chosen hierarchically.
5. The best USP link cost setting for each network according to $\rho_{\mathcal{L},OFIB}^{\max}$, abbreviated as O. In case several link cost settings leading to the same best value exist, the one with the lowest values of $\rho_{\mathcal{L},FRR}^{\max}$, $\rho_{\mathcal{L}}^{\max}$, and ρ_{\emptyset}^{\max} is chosen hierarchically.

3 Analysis of Optimization Effectiveness and Extension to the Full Failure Cycle

Figure 3.20(a) compiles for each network and for each of the chosen link cost settings the maximum relative link load values for the different full failure cycle phases. The topologies are ordered alphabetically. Figures 3.20(b) and 3.20(c) enlarge two detailed views of the figure that will be addressed in the following paragraphs.

The solid bars correspond to the maximum relative link load values actually reached by a certain link cost setting. The dashed bars indicate how bad the maximum relative link load values can become in phases that have not been optimized. This is explained for the best link cost setting according to $\rho_{\mathcal{L}}^{\max}$ in the COST network, see Figure 3.20(b). The yellow bar corresponds to the best possible $\rho_{\mathcal{L}}^{\max}$ value found in the database. The red bar shows, for all the link cost settings in the database leading to that best $\rho_{\mathcal{L}}^{\max}$ value, the best ρ_{\emptyset}^{\max} value that any of these link cost settings leads to. When optimizing $\rho_{\mathcal{L}}^{\max}$, the maximum relative link load during the NotVia FRR phase is not regarded and can become arbitrarily bad. Therefore, the dotted green bar shows the worst $\rho_{\mathcal{L},FRR}^{\max}$ value found in the database for all link cost settings leading to the best $\rho_{\mathcal{L}}^{\max}$ value. This is not necessarily the same link cost setting leading to the best ρ_{\emptyset}^{\max} value shown by the red bar and therefore displayed as a dotted not solid bar only. Analogously, for each regarded class of link cost settings, H, \emptyset , \mathcal{L} , F, and O, all values included in the optimization are shown as solid bars. For all other values, worst cases are indicated as dotted bars. In the particular case of HC routing, there is only a single link cost setting (all link costs set to 1) and therefore, all bars are plotted in solid.

Based on this illustration, some of the most important findings concerning link cost optimization are summarized.

It can be seen from Figure 3.20 that unoptimized HC routing can be arbitrarily bad. As the values are based on the same ECMP HC traffic matrix as used before, in particular, values larger than 100% are possible. This is due to the fact, that SSP routing is considered.

The optimization potential varies for different topologies. However, for all considered networks, significant improvements compared to unoptimized HC routing are possible.

The more phases of the full failure cycle are included in the optimization, the more of the different maximum relative link load values can be reduced. This can be seen, e.g., in Figure 3.20(c). If only $\rho_{\mathcal{L}}^{\max}$ is optimized, the values during the NotVia FRR phase can be very high. If $\rho_{\mathcal{L},FRR}^{\max}$ is included in the optimization, it can be reduced to the same value as $\rho_{\mathcal{L}}^{\max}$. However, in this case the temporary load increase caused by OFIB can still be significant. If all phases are included in the optimization, $\rho_{\mathcal{L},OFIB}^{\max}$ can be reduced to the same value as $\rho_{\mathcal{L}}^{\max}$ as well. In this case, this does not cause an increase of the corresponding ρ_{\emptyset}^{\max} or $\rho_{\mathcal{L}}^{\max}$ values. The latter does however not hold for all topologies. If more phases are included in the optimization, the objective is to minimize the maximum relative link load considering all these phases. Consequently, link cost settings have to be found that lead to a routing not exceeding this maximum relative link load value in any of the considered phases. This however, with each additionally considered phase, puts more constraints on the routing layout.

Layouts leading to better values when considering only a smaller set of phases might not be possible anymore when the maximum relative link load in additional phases has to be considered. Therefore, the best values obtained when optimizing for a smaller set of considered phases might not be reachable anymore if more phases are regarded. An example can be seen for the COST network in Figure 3.20(b). The link cost settings in the database leading to the best $\rho_{\mathcal{L},FRR}^{\max}$ value do not contain any routing leading to a $\rho_{\mathcal{L}}^{\max}$ value as good as if $\rho_{\mathcal{L},FRR}^{\max}$ is not considered, see left and middle yellow bars. Other examples are, e.g., the GE, LA, and SP network, where the best link cost settings according to $\rho_{\mathcal{L}}^{\max}$ lead to a much higher ρ_{\emptyset}^{\max} value than link cost settings only optimized for ρ_{\emptyset}^{\max} . This can also be due to the fact, that the best values according to ρ_{\emptyset}^{\max} have been chosen from those settings in the database being USP in the failure-free case. The best values according to $\rho_{\mathcal{L}}^{\max}$ from those being USP also in all single link failures \mathcal{L} .

Finally, for all networks considered, the last stacked bars, i.e., the ones corresponding to the best $\rho_{\mathcal{L},OFIB}^{\max}$ link cost settings do not show any violet OFIB bars. Thus, as already shown before, the temporary load increase due to OFIB can be eliminated in all considered networks when regarded during the optimization.

3 Analysis of Optimization Effectiveness and Extension to the Full Failure Cycle

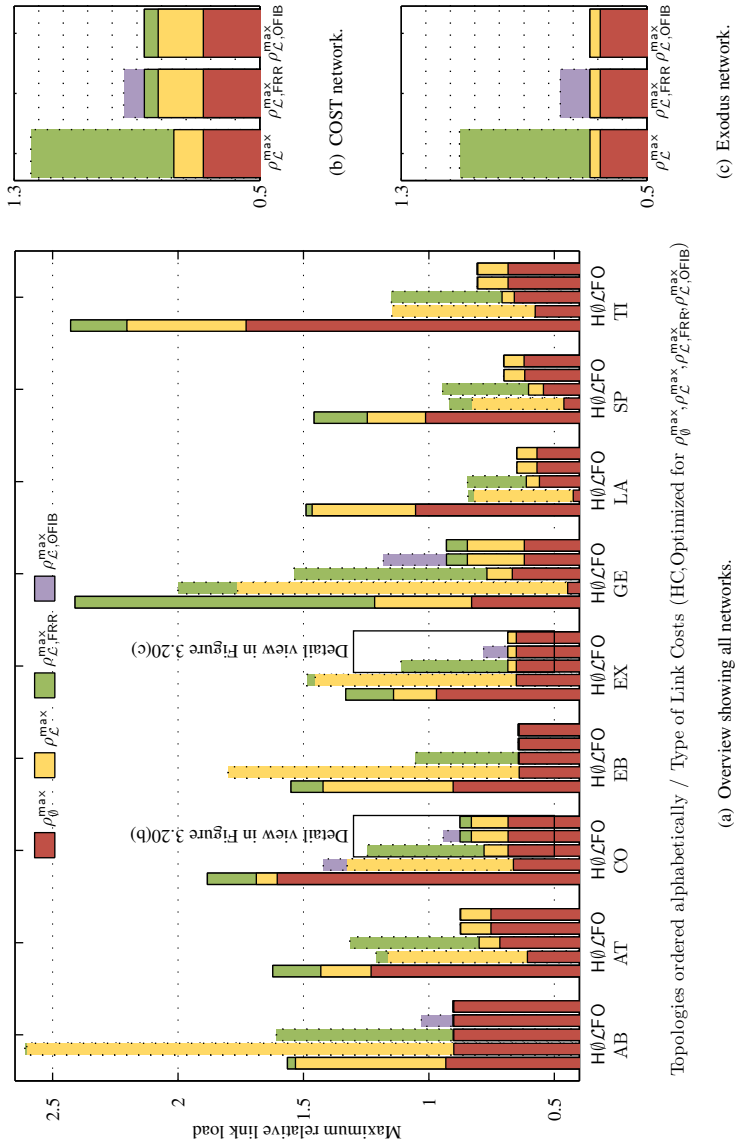


Figure 3.20: Visualization of the effect of full failure cycle optimization.

3.3 Related Work

An overview on related work on IP routing optimization has already been given in Chapter 2. In Chapter 2, resilience has been regarded rather implicitly by analyzing and optimizing the routing not only for the failure-free case but also for a set of failures and considering FRR mechanisms. In the analysis framework used in this chapter, resilience is explicitly addressed and measured. Therefore, here, related work on resilience in general and resilience analysis is presented. Furthermore, related work on LFC and OFIB is provided.

3.3.1 Resilience and Resilience Analysis

Resilience in general is a very wide research field and covers many different aspects. These include different resilience mechanisms, see e.g. [125], resilience optimization as e.g. regarded in this monograph, in routing optimization in general, or in network design [126], and resilience analysis which is focused in this description of related work. Other terms related to resilience found in literature include *reliability*, *fault tolerance*, *robustness*, *availability*, *reachability*, and many others. Sterbenz et al. [127] provide a classification of different *resilience disciplines* and propose a way how all different terms related to resilience can be interconnected.

A network can be resilient (or tolerant) against different effects including according to [127], e.g., failures, disruptions, or changing traffic. The resilience analysis framework used here [39,40] mainly covers the aspects *fault tolerance*, *disruption tolerance*, and *traffic tolerance*.

Resilience can be regarded on different (technical) levels or layers. In the discussions in this chapter, the resilience of an IP network is regarded. However, the used analysis framework can also be applied to any other type of packet-switched communication network. Other works regard, e.g., multi-layer resilience [126–129] or resilience in virtual networks [130–132].

An important issue connected to resilience analysis is the choice of the used method or metric to quantify resilience. Menth et al. [39,40] provide an overview on different methods in literature to measure resilience and propose their own method based on probabilities for certain failures and traffic surges which is used in the studies presented in this chapter. The focus of the presented metrics is, as mentioned above, on analyzing the aspects *fault tolerance*, *disruption tolerance*, and *traffic tolerance*. More particular, as described before in Section 3.1, the evaluations presented in this chapter focused on the risk of overload in case of failures appearing with a probability at least as high as a preconfigured threshold. Two mapping functions were used, one based on overload probabilities, the other based on relative link load quantiles. Further metrics for resilience are discussed, e.g., in [133–136] and the references within. These metrics contain among others availability measures for a certain link, path, or set of paths or measures considering the impact caused by certain failures. Some of the metrics are only based on topological properties of a topology, others involve path layouts or are additionally weighted with the amount of traffic of particular considered demands.

3.3.2 Loop-Free Convergence and Ordered FIB Updates

To the best knowledge, there is only few related work concerning LFC and OFIB. It is briefly described in the following.

Goyal et al. [137] provide a survey on "Improving convergence speed and scalability in OSPF". They explain the procedures involved in the convergence process and give an overview on possible speedups. These include among others a faster failure detection, more efficient communication between adjacent nodes, faster *Link State Advertisements (LSAs)* for failure notifications, faster routing table calculation or a graceful restart of the control plane of a node without influencing the forwarding on its data plane. The authors also address FRR mechanisms and LFC as so-called *proactive approaches to failure recovery*.

The RFC5715 [138], based on the analysis of Zinin [139], discusses the causes for micro-loops in general, gives an overview on counter measures to micro-loops

and discusses how the number of micro-loops can be minimized. Four *micro-loop control strategies* are classified that shall be only briefly mentioned here. i) *Micro-loop mitigation* tries to reduce the formation of possible micro-loops by network reconfiguration but does not guarantee to eliminate all of them. ii) *Micro-loop prevention* proposes mechanisms to prevent all micro-loops of appearing. Examples for such mechanisms are, e.g., *incremental cost advertisement* as proposed in [140, 141], *packet marking* as e.g. used in [142], or *ordered FIB updates*, i.e., the OFIB concept discussed here. iii) The third class stated in RFC5715 [138], *micro-loop suppression* does not try to eliminate micro-loops but the "collateral damage caused by micro-loops to other traffic" e.g. by monitoring and dropping looping packets. iv) The last classified control strategy is a *network design to minimize micro-loops*, where possible micro-loops need to be considered and avoided already during the network design process.

Francois et al. [124, 143] show that micro-loops may occur during the convergence of link state routing protocols depending on the update order. Furthermore, they introduce the OFIB concept and thereby show that it is possible to define update orders that effectively avoid micro loops. Finally, they show by simulations that sub-second LFC is possible on a large Tier-1 Internet Service Provider network. Relative link loads during OFIB are not considered in these papers. Fu, Shi et al. [144, 145] address the problem of temporary load increase during the LFC phase. They propose to tackle this issue by calculating special update orders that reduce the load increase. The basic idea is to always reroute the flow that causes the least overload. Shi et al. [145] extend the idea of Fu et al. [144] by assuming that the nodes do not need to update their entire forwarding table en block but that they can do partial updates of flows one by one. This heuristic requires several iterations that leads to the proposed algorithm being slower than the one of Fu et al. [144]. The methods of both papers have been tested in example networks for a number of failure scenarios. Both papers revealed that modifying the update orders brings no guarantee to avoid temporary load increase during the LFC phase. In this chapter, it is not tried to improve the update order but to optimize administrative link costs in such a way that the temporary load increase is avoided independent of any particular update order.

3.4 Lessons Learned

This chapter focused on the analysis of the effectiveness of link cost optimization and on the extension of the optimization to the full failure cycle. Two issues were addressed: i) how do routing layouts optimized only for a small set of failures perform in more severe failure scenarios not regarded during the optimization and ii) how can the *Loop-Free Convergence (LFC)* phase of link state routing be included in the link cost optimization?

To make the optimization of resilient routing computationally feasible only a rather small set of the most probable failures can be regarded. In the link cost optimization with the NetOpt tool used in this monograph, usually all single link failures are regarded. This fact leads to the question how resilient link cost settings optimized only for single link failures are when regarding more severe failures such as the simultaneous outages of several links. To address this question, the resilience analysis framework proposed in [39,40] was used to analyze the effectiveness of the link cost optimization with the example of the Exodus network already regarded in Chapter 2. Resilience analysis evaluates the load conditions in communication networks for a large set of likely failure scenarios \mathcal{Z} whose probabilities are at least p_{min} . In the presented example analysis, p_{min} was set to a value of 10^{-15} , resulting in a number of $|\mathcal{Z}| = 51577$ considered scenarios including the failure patterns $\emptyset, L, N, LL, LN, NN, LLL, LLN$, where L denotes a single link and N a single node failure. As mentioned before, routing optimization is usually applied to improve load conditions only for a set of most likely failure scenarios \mathcal{X} which is up to a thousand times smaller than \mathcal{Z} . In case of the Exodus network the failure-free case \emptyset as well as all single link failures $\mathcal{X} = \mathcal{L}$ with $|\mathcal{L}| = 51$ were regarded. Despite of this large difference in size of the considered failure sets, it was shown that routing optimization significantly reduces potential overload in networks with conventional IP routing and rerouting.

Different mapping functions were looked at that map for each link the *Complementary Cumulative Distribution Function (CCDF)* of the relative link loads

in different failure scenarios to a single value. Regarding the mapping value $R_q^{0.99999999}$ corresponding to the 99.999999% quantile of the CCDF of the relative link load, it was shown that for the Exodus network this 99.999999% quantile rises to up to a value of about 116% for single links in the network in case of unoptimized HC routing. In the given example, USP routing optimization allows to reduce the worst $R_q^{0.99999999}$ value of any link to about 72%. Routing optimization does not reduce the traffic but only distributes it to other links in the network. Therefore, as traffic previously on the links with the highest risk is redistributed in the network, this leads to a light increase of the $R_q^{0.99999999}$ values of other links. However, the analysis showed that this increase does not cause any real problems because the relative link loads still remain relatively low.

Regarding the example of unoptimized and optimized NotVia, it was illustrated that without routing optimization, *IP Fast Reroute (IP-FRR)* possibly causes even more overload than conventional routing and rerouting. The worst $R_q^{0.99999999}$ value in case of unoptimized NotVia HC routing in the Exodus network was obtained as about 130%. However, routing optimization is again very effective in avoiding potential bottleneck situations and reduces this worst $R_q^{0.99999999}$ value to about 85%. This shows that routing optimization is even more beneficial in this case due to the higher risk of overload in the unoptimized case. Moreover, as already known from Chapter 2, it is needed for IP-FRR anyway because the link cost values should be chosen in such a way that equal-cost paths are avoided in order to obtain unambiguous backup paths. The ResiLyzer tool implements the used resilience analysis framework. In this chapter, it was briefly presented and extended by further visualizations, such as adjacency matrix illustrations and CCDFs of mapping function values.

In the second part of this chapter, the inclusion of the LFC phase in the IP routing optimization and the extension of the optimization to the full failure cycle was investigated. Therefore, first an overview on the *Ordered FIB Updates (OFIB) Loop-Free Convergence (LFC)* mechanism was provided. It is deployed to avoid packet loss due to micro-loops during the reconvergence after a failure. Nevertheless, unanticipated temporary increases of the relative link load that can also

lead to packet loss can still occur in this phase. It was demonstrated that the actual relative link load depends on the router update order, which is only partially specified by OFIB. As the number of OFIB conform update orders increases exponentially with the number of nodes in the network, an evaluation of all possible update orders is computationally not feasible. Therefore, to estimate the possible temporary load increase during the OFIB phase, for each considered network and routing a number of 1000 random update orders was evaluated. The evaluations revealed that some link loads can temporarily exceed the maximum relative link load that occurs otherwise in the network, including FRR and reconverged routing after failures. This effect can be significant and lead to the conclusion that the LFC phase has to be considered in the link cost optimization process.

As an evaluation of all update orders is not feasible, a simple and fast mechanism was provided to calculate a tight upper bound on the relative link load increase. Evaluations showed that in about 90% of all considered cases, the upper bound is equal to the worst value found when evaluating 1000 random update orders. In more than 99% of the cases, the overestimation of the upper bound compared to the worst found value is less than 10%.

Using the proposed upper bound, the routing was optimized for multiple network topologies already presented in Chapter 2 to minimize the maximum relative link load in the failure-free state, the FRR state, and the reconverged state with and without inclusion of all OFIB stages. The inclusion of OFIB states in the optimization for all considered topologies completely avoided the effect of temporary load increases on the maximum relative link load and did not lead to worse relative link load values in the other routing stages.

Hence, a relative link load analysis covering all possible routing states during failure handling and recovery was performed and provided a resilient IP routing optimization that considers all of these states. The chapter also provided a summary of the link cost optimization by comparing the maximum relative link load during different phases of the full failure cycle for different degrees of optimization. The results showed that link cost optimization offers a high potential to improve the routing layout. In some of the considered topologies, the maximum

relative link load could be decreased by up to 30% compared to the unoptimized HC routing even when including the resilient routing in case of outages, IP-FRR, and the OFIB phases.

4 Resilience Enhancements for the Control Plane of SDN-based Core Networks

In this chapter, resilience enhancements for the control plane of SDN-based core networks are discussed. The recent introduction of *Software Defined Networking (SDN)* has caused a paradigm shift in communication networks. The SDN concept allows separating network functions in control plane and data plane, namely moving complex functions from devices in a network to sophisticated dedicated controller instances. In the particular case of routing and forwarding, SDN allows to handle all routing issues inside central control units while normal switches only forward packets according to the rules provided by the controllers.

The most popular implementation of the SDN concept is OpenFlow [38], where a central OpenFlow *controller* defines rules for switches how to handle packets, thus enabling a centralized routing approach. OpenFlow basically offers possibilities to base a forwarding rule for a flow on any TCP/UDP+IP header field so that no predefined labels as in *Multiprotocol Label Switching (MPLS)* are necessary but packets can be filtered and forwarded according to any desired information. In case a switch already has a forwarding rule matching a given packet, the packet is processed according to this rule. In case a switch does not know what to do with a packet, it will - in default configuration - contact its assigned controller and ask for a rule what to do. That way, routing can be even configured and modified in a running system.

With the HyperFlow [146] concept, OpenFlow networks can be separated into several domains, each with their own controller. This facilitates load balancing and resilience features in the SDN infrastructure.

Considering the possible routing layouts, SDN can be used to implement any layout that could be configured with MPLS as described, e.g. in Chapter 2. As already mentioned above, SDN even offers much more possibilities for path layouts due to the different way of operation. Whereas in IP networks forwarding is done according to destination addresses and in MPLS networks based on the configured labels of a packet stream, in SDN, forwarding can be based on any header information of a packet. Therefore, in particular, the optimized path layouts discussed in the previous chapters can also be applied to SDN-based networks and are not further discussed in the following.

The concept of a logically centralized control plane, separated from the distributed data plane, however, poses new resilience questions that are investigated in this chapter. Two crucial issues for external control architectures are analyzed: i) how many controllers are necessary in a network to be resilient against the most typical outages in a network and ii) where to place them for an adequate trade-off between latency and resilience?

Heller et al. [147] indicated that the topic of general controller placement is well explored and no new *theoretical* insights are expected. In particular, the very basic version of controller placement according to the latency of nodes to their controller, also known as *facility* or *warehouse location problem*, is a typical example for a *Mixed Integer Linear Program (MILP)* provided e.g. with the *IBM ILOG CPLEX* [148] software. Much work on the topic of controller placement in literature concentrates on the fact that the problem is NP-hard and depending on the complexity of the particular considered objective often provides only approximations to solve it. Heller et al. showed that finding optimal solutions is computationally feasible for realistic network instances and failure-free scenarios, by analyzing the entire solution space using "weeks of computations" on today's CPUs. Thus, they could address and optimally answer the question posed before but without considering failure tolerance. They revealed that in

most topologies one single controller is enough to fulfill "existing reaction-time requirements". In this chapter, the controller placement analysis of Heller et al. is extended to include different resilience aspects that are important in the context of SDN. In particular, it is shown that in most topologies, where a single controller would be enough from a latency point-of-view, many more controllers are necessary to meet resilience requirements. Also inter-controller latency, load balancing between controllers, and trade-off considerations between latency and failure resilience are taken into account. A Matlab-based framework to compute resilient Pareto-based Optimal COntroller-placements called *Pareto-based Optimal COntroller-placement (POCO)* is introduced. It makes an efficient combined use of CPU and RAM, so it can evaluate the entire solution space even when resilience is considered. The advantage of this approach compared to any particular MILP or heuristic is that evaluating the entire solution space naturally provides information for all considered objectives for all placements. Regarding multi criteria/multi objective optimization, that means that no decision on the importance of certain objectives has to be taken before invoking the optimization by defining some constraints or weighted objective functions. In contrast, offering all possible solutions evaluated by all objectives, offers the possibility to take the decision afterwards. In literature, there also exist different approaches for multi criteria facility location for a given combination of certain objectives, see e.g. [149–154] and references within. However, the approach of evaluating the entire solution space offers – for realistic network sizes – the most freedom to consider different objectives. POCO does *not* offer a recipe to solve all instances of the problem of any size, but it can find optimal solutions for realistic network sizes. The POCO-toolset is able to solve the problems within acceptable computation time.

The content of this chapter is mainly taken from [16]. The remainder of this chapter is organized as follows. In Section 4.1, a brief introduction to SDN as well as different SDN scenarios is given, and different challenges with central controllers and how good placements can alleviate or avoid these problems are illustrated. Section 4.2 addresses the optimization of resilient controller placement including the resilience against different possible network failures (Sec-

tion 4.2.1), load balancing among different controllers and how to account for inter-controller latencies (Section 4.2.2). Then, it briefly introduces POCO in Section 4.2.3 and the calculation using *MATLAB* matrix operations. Section 4.3 gives an overview on related work. The chapter is concluded and the results are summarized in Section 4.4.

4.1 Scenarios and Problem Description

This section briefly explains SDN and different SDN scenarios. Afterwards, several aspects that have to be considered for a resilient controller placement in SDN networks are shown.

4.1.1 SDN and SDN Scenarios

One of the key ideas of Software Defined Networking is to separate the network control plane. Functionality, such as routing, is taken out of the data plane, i.e. it is taken away from normal network nodes, and moved towards external controllers. Depending on the particular use case, these controllers can either be realized in hardware or as pure software components. A single controller or a set of controllers communicating with each other may exist. Controller communication architectures can be either flat, with each controller having the same role, hierarchical with normal controllers and coordinating "master controllers", or follow a variety of different approaches. The connection between the nodes and their controllers can be realized inband or outband, i.e. using the same physical connections or dedicated lines.

The concept of SDN can be used for a variety of use-cases. Figure 4.1 illustrates different possible SDN use cases in future networks. SDN can be, among others, used to control the monitoring, the traffic and to achieve load balancing in data centers. Furthermore, it can be used for traffic engineering purposes in access networks, or to setup and run a unified logical business network consisting of physically distributed sub networks. The idea of *Network Functions Virtualization (NFV)* is to virtualize different network functionalities, such as monitoring,

firewalling, or security aspects. SDN can serve as an enabler for this approach, e.g. by providing certain functionalities on central controllers.

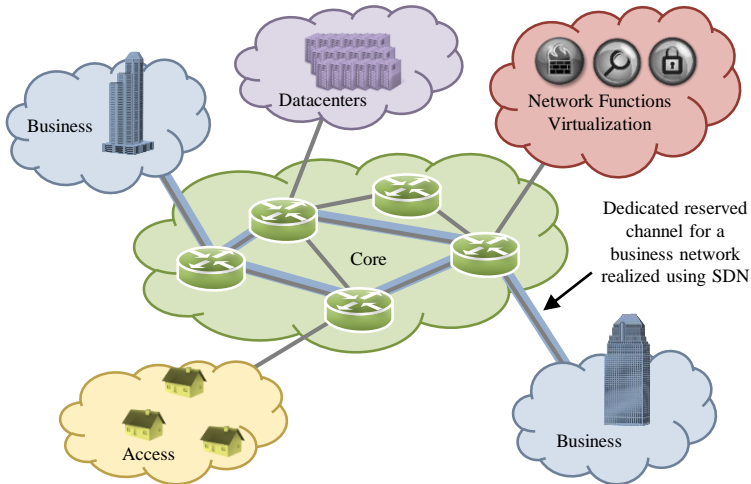


Figure 4.1: *Inter-connection of different possible use cases for SDN. This chapter focuses on the use case of SDN in core networks as well as a possible extension towards NFV.*

This chapter focuses on the use case of SDN in core networks as well as a possible extension towards NFV. It is assumed that usually in core networks primary paths are pre-installed for traffic aggregated between different nodes in the network similar to pre-installed MPLS labels. SDN controllers are thus not being contacted for each single flow, but only in case of traffic engineering actions or in case of outages to find adequate backup paths. Therefore, depending on the network size a single controller might be able to control the entire network without being overloaded. However, for resilience issues, more than one controller is necessary. If NFV functionalities are based on an SDN control platform, the

number of necessary controllers is significantly higher due to the heavy load on the control plane. It is assumed that in SDN core networks, controllers are co-located with regular network nodes. The signaling between nodes and controllers is done in-band, in the sense of "in the same physical network". That means that if the network is physically disconnected in several parts, nodes and controllers in different parts of the network cannot contact each other anymore.

4.1.2 Resilient Controller Placement for SDN

A main objective for a good controller placement is to minimize the latencies between nodes and controllers in the network. However, looking only at delays is not sufficient. A controller placement should also fulfill certain resilience constraints. To illustrate this, the best controller placement with $k = 5$ controllers in the Internet2 *Open Science, Scholarship and Services Exchange (OS³E)* topology according to maximum latency as shown by Heller et al. [147] is examined. Figure 4.2 shows four different illustrations of the same placement to depict potential problems to be considered when judging the resilience of a placement. In the following, these issues are briefly explained as well as what is necessary to be resilient against them.

Controller Failures

As illustrated by Heller et al. [147], a larger number of well-distributed controllers in a network can obviously help to lower the maximum latency between the nodes and their controllers. It also increases the failure tolerance if some of the controllers stop working. Zhang et al. [155] assume in their work that a node is not able to route anymore if it loses its connection to the controller. Here, it is supposed that in case of a controller outage, it is possible to reassign all nodes previously attached to that controller to their second closest controllers in the network using a backup assignment or signaling based on normal shortest path routing.

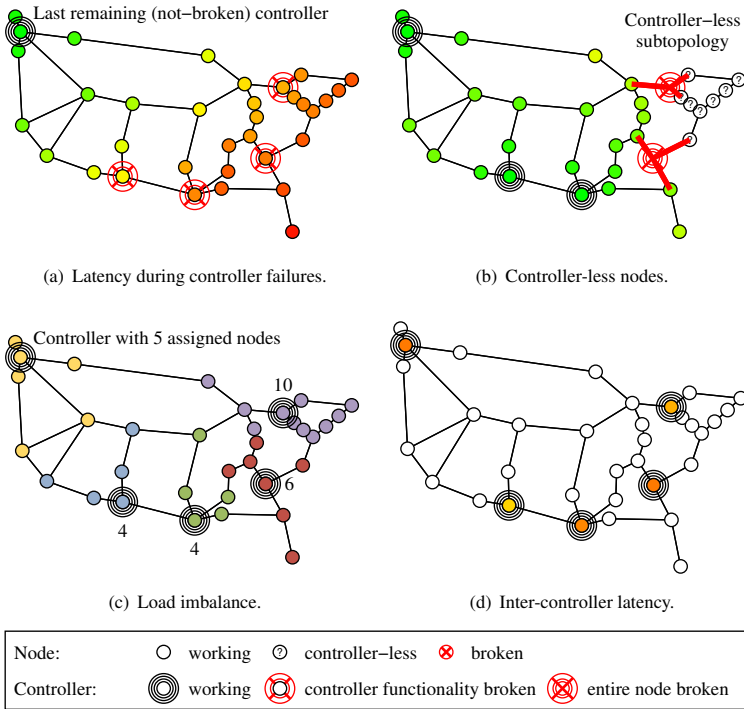



Figure 4.2: Illustration of different issues to be considered when judging the resilience of a controller placement.

Thus, as long as at least one of the controllers is still reachable, all nodes remain functional. However, the latencies of the reassigned nodes to their new controller can be significantly higher than the latencies to the primary controller. Figure 4.2(a) illustrates the latencies of all nodes to the last remaining controller in case of an outage of all other four controllers using a traffic light color scheme. The color changes from pure green indicating a latency of zero to yellow indicat-

ing 50% of the network's diameter to pure red indicating 100% of the diameter. The displayed controller failure scenario corresponds to the worst case since the remaining controller is the most remote one. Thus, the requests of some of the nodes need to pass almost through the entire network to reach the controller. To increase resilience against this phenomenon, the controller placement optimization should not only consider the latencies during failure-free routing, but also worst case latencies during controller failures. This problem is addressed in Section 4.2.1.

Network Disruption

In contrast to controller failures, the outage of network components, such as links and nodes, often has a much higher impact on the network stability, as it alters the topology itself. The shortest paths between some of the nodes change, leading to different latencies and possibly to the reassignment of nodes to other controllers. Even more severe is that entire parts of the network are in danger of being cut off by link or node outages. In the worst case, some nodes can no longer be connected to a controller as they are cut off from all controllers. These nodes are still working and able to forward traffic, but cannot request instructions anymore. Figure 4.2(b) illustrates the worst possible scenario for double node failures. All nodes depicted with a white question mark icon  are controller-less, i.e. still working but cannot reach any controller. Hence, the entire subnetwork consisting of these controller-less nodes is no longer able to address any functionality realized by the controller despite the fact that the nodes are still working. Rerouting flows to working paths is no longer possible, even though some of the nodes are still physically connected. In Section 4.2.1, the problem of controller-less nodes and network component failure tolerance is addressed. Among others, it is illustrated why $k = 5$ controllers are not enough in the Internet2 OS³E topology to avoid controller-less nodes in all possible double link and node failure scenarios. Furthermore, the minimum number k of controllers to eliminate the occurrence of controller-less nodes for these failure scenarios is calculated.

Load Imbalance

Analog to [147], it is assumed that nodes are always assigned to their nearest controller using latency as metric. The latency is computed as the shortest path $d_{n,c}$ between the node n and controller c with the cost of each link of the network being equal to the geographical distance between the two cities containing the nodes connected by this link. Regarding the fact that in the following discussion this geographical distance is given in a precision of meters, the probability that the length of two paths in the network is exactly identical can be neglected. Therefore, according to the knowledge from Chapter 2, one can assume that all shortest paths between any two nodes and thus the node-to-controller assignment are unique. Figure 4.2(c) uses different colors to illustrate the node-to-controller assignment. The number of nodes per controller is imbalanced and ranges from 4 to 10. The more nodes a controller has to control, the higher is the load on that controller. This is especially relevant in scenarios where nodes communicate often with their controller, e.g. when considering NFV. If the number of node-to-controller requests in the network increases, so does the chance of additional delays due to queuing at the controller system.

To be resilient against controller overload, the assignment of nodes to the different controllers should be well-balanced. In [156], it was demonstrated that controller performance can vary among connected switches. This highlights the importance of an intelligent controller placement that also takes load balancing aspects into account. This issue is addressed in Section 4.2.2.

Inter-Controller Latency

It is clear that a single controller is not enough to reach any kind of resilience in a network. However, when several controllers are placed in the network, another issue arises that is briefly addressed in Section 4.2.2. If the control logic of the network is distributed over several controllers, these controllers need to synchronize to maintain a consistent global state. Depending on the frequency of the inter-controller synchronization, the latency between the individual con-

trollers plays an important role. Figure 4.2(d) illustrates the maximum controller-to-controller latencies using the same traffic light color scheme relative to the network diameter as in Figure 4.2(a). The color of each controller indicates the maximum distance of this controller to all others. For the depicted placement, the messages between the controllers have to travel relatively long distances in the network which might not be acceptable.

4.2 Optimization of Resilient Controller Placement

This section addresses the optimization of resilient controller placement according to the afore mentioned issues including the resilience against different possible failures (Section 4.2.1), load balancing among different controllers (Section 4.2.2) and how to account for inter-controller latencies (Section 4.2.2). Then, it introduces POCO and the calculation implemented in it using *MATLAB* matrix operations.

4.2.1 Failure-Tolerant Controller Placement

In this section, resilience against different failures is included into the controller placement and optimal results are calculated using the POCO-framework. First resilience against controller failures is discussed, then resilience against network element failures.

Controller Failure Tolerance

When a controller fails, some nodes are reassigned to other controllers and experience increased latencies. To quantify the latencies in a network, the maximum over all node-to-controller-latencies is considered and denoted with $\pi^{\max \text{ latency}}$. Similar as presented in [147], based on a matrix $d_{v,w}$ containing the shortest path distances between all nodes v and w of the set of all nodes \mathcal{V} , the maximum

node-to-controller latency for a placement of controllers \mathcal{P} , being a non-empty subset of the power set $2^{\mathcal{V}}$, can be defined as

$$\pi^{\max \text{ latency}}(\mathcal{P}) = \max_{(v \in \mathcal{V})} \min_{(p \in \mathcal{P})} d_{v,p}. \quad (4.1)$$

Here, the maximum latency is considered instead of the average latency, because an average hides the worst case values that are important when resilience should be improved. All latency values are calculated relatively to the diameter of the network in the failure-free case (% of diameter). In the failure-free case, the maximum latency is denoted as $\pi_{\emptyset}^{\max \text{ latency}}$. For a placement of k controllers, a set of scenarios \mathcal{C} is constructed that includes all possible combinations of up to $k - 1$ controller failures (including the failure-free case \emptyset) and the resulting maximum latency is denoted with $\pi_{\mathcal{C}}^{\max \text{ latency}}$.

$$\pi_{\mathcal{C}}^{\max \text{ latency}}(\mathcal{P}) = \max_{(s \in \mathcal{C})} \pi_s^{\max \text{ latency}}(\mathcal{P}). \quad (4.2)$$

The node-to-controller assignments change in case of controller outages. Therefore, for the metric $\pi_{\mathcal{C}}^{\max \text{ latency}}$, not only the distance to the (primary) controller in the failure-free case but also the distance to the other (backup) controllers in case of failures is included in the metric. Let the placements \mathcal{P}_1 be all these (non-empty) subsets of working controllers of a placement \mathcal{P} that can appear for the considered controller failure scenarios \mathcal{C} . Then, $\pi_{\mathcal{C}}^{\max \text{ latency}}$ can also be obtained as

$$\pi_{\mathcal{C}}^{\max \text{ latency}}(\mathcal{P}) = \max_{(v \in \mathcal{V})} \max_{(\emptyset \subset \mathcal{P}_1 \subseteq \mathcal{P})} \min_{(p \in \mathcal{P}_1)} d_{v,p}. \quad (4.3)$$

When controllers are placed in the network so that $\pi_{\emptyset}^{\max \text{ latency}}$ is minimized, the intuitive result is a placement where controllers are equally spread in the network. On the other hand, when placing several controllers to reach the best possible $\pi_{\mathcal{C}}^{\max \text{ latency}}$ even in the worst failure cases, all controllers tend to be in the center of the network. Thus, even if all except for one controller fail, the

latencies are still satisfying. Figure 4.3 shows the optimal placement $\mathcal{P}_C^{\max \text{ latency}}$ for $k = 5$ controllers. The corresponding minimal maximum latency is denoted as $\underline{\pi}_C^{\max \text{ latency}}$. For a given number of k controllers, it can be obtained as:

$$\underline{\pi}_C^{\max \text{ latency}} = \min_{\mathcal{P} \in \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_{\binom{|V|}{k}}\}} \pi_C^{\max \text{ latency}}(\mathcal{P}). \quad (4.4)$$

As expected, in the depicted case, the controllers are all situated close to the center of the network. Even in the worst case situation shown in Figure 4.3, where all but one controller fail, the maximum latencies are still relatively low. This particular placement with $k = 5$ controllers actually corresponds to a combination of the five best placements of single controllers. The maximum latency can obviously never decrease with an increasing number of failed controllers. The worst case, i.e., the highest latency, is reached when 4 out of 5 controllers fail and only the fifth best single controller placement is still active. In general, when considering up to $k - 1$ simultaneous controller failures, i.e., in particular, all subsets $\mathcal{P}_1 \subset \mathcal{P}$ containing only a single working controller, the value of $\pi_C^{\max \text{ latency}}$ corresponds to the maximum distance of all nodes to their k -closest controller in the failure-free case. Therefore, another shorter and easier to calculate definition of $\pi_C^{\max \text{ latency}}(\mathcal{P})$ can be given for this case:

$$\pi_C^{\max \text{ latency}}(\mathcal{P}) = \max_{(v \in V)} \max_{(p \in \mathcal{P})} d_{v,p}. \quad (4.5)$$

Even though the maximum latencies are still relatively low, even in the case of $k - 1$ simultaneous controller outages, this centralized placement with low $\pi_C^{\max \text{ latency}}$ leads to an increase of $\pi_{\emptyset}^{\max \text{ latency}}$ compared to the best placement $\mathcal{P}_{\emptyset}^{\max \text{ latency}}$ optimized for the failure-free scenario with an equal number of controllers.

There is a clear trade-off between the placements optimized for the failure-free case and those including controller failure resilience. To look at this trade-off in more detail, all possible placements with $k = 5$ controllers are shown in Figure 4.4 with their quality according to the failure-free and the resilient case for the metric $\pi^{\max \text{ latency}}$. Each point in the graph indicates one placement. The

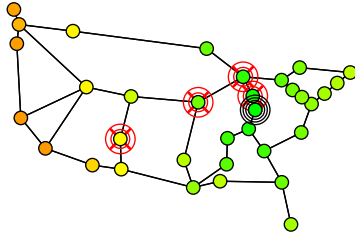


Figure 4.3: Impact of worst case failure scenario (4 out of 5 controllers fail) on node-to-controller latency.

x-value of a point indicates the $\pi_{\theta}^{\max \text{ latency}}$ value of the corresponding placement, the y-value the $\pi_C^{\max \text{ latency}}$ value. As mentioned before, the outage of up to all but one controller is regarded. The dashed lines indicate the mean values for the corresponding metric over all placements.

For better visibility, the axes limits of the graph have been adapted to display only the most important range of the placement solution space. To give an impression, the worst placements have values of $\pi_{\theta}^{\max \text{ latency}} > 85\%$ of the diameter.

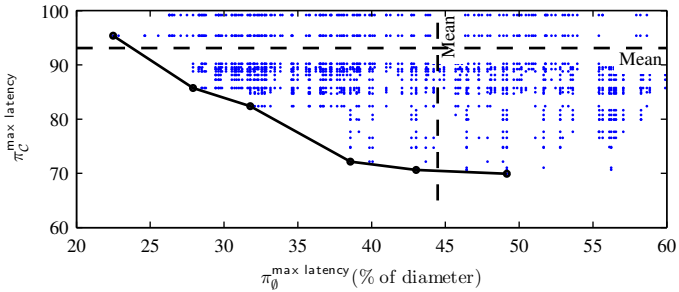


Figure 4.4: Trade-off between failure-free and controller failure values.

The black points connected with a black line indicate the set of Pareto-optimal placements which is returned by the POCO-framework out of all possible placements. For a number of n metrics π_1, \dots, π_n , a value (x_1, \dots, x_n) is Pareto-optimal if and only if there is no other value (y_1, \dots, y_n) with y_i better or equal to x_i for all metrics π_i and y_j strictly better than x_j for at least one metric π_j . In Figure 4.4, there is no single optimal placement with best possible values for both the failure-free case and the controller failure case. In contrast, Pareto-optimal placements that perform better in the resilient case perform worse in the failure-free case and vice versa. Thus, POCO usually gives no recommendation for a particular placement, but returns the set of Pareto-optimal placements, which enables the network operators to choose the placement that fits best to their needs. In particular, they can also decide up to how many controller failures should be covered by a resilient placement.

To verify that the observation also holds for other topologies, the trade-off between the optimal placements $\mathcal{P}_\emptyset^{\max \text{ latency}}$ and $\mathcal{P}_C^{\max \text{ latency}}$ is evaluated for all 146 topologies¹ in the Topology Zoo collection [157] with a size of at least 5 and up to 50 nodes for $k = 5$ controllers. Figure 4.5 shows the distribution of the worst values $\pi^{\max \text{ latency}}$ for the optimal placements $\mathcal{P}_\emptyset^{\max \text{ latency}}$ (solid lines) and $\mathcal{P}_C^{\max \text{ latency}}$ (dashed lines) over all topologies for 0, 2, and 4 failures respectively (from left to right). The results confirm the previous findings. Without failures (green lines), $\mathcal{P}_\emptyset^{\max \text{ latency}}$ achieve the lowest delays, as they are optimized for this case. But when four controllers break down simultaneously (red lines), such placements lead to high latencies in a very large fraction of the examined topologies. As expected, the placements $\mathcal{P}_C^{\max \text{ latency}}$ that are optimized for up to four controller failures provide far lower latencies. But also when only two out of five controllers fail (blue lines), the controller placements that are optimized for up to four failures yield lower latencies than the placements optimized for the failure-free case.

¹If several versions of the same topology exist, only the most recent one was considered.

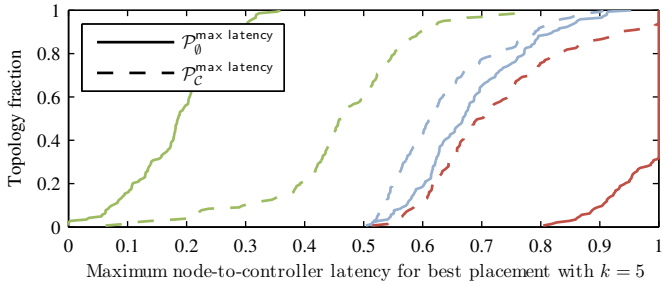


Figure 4.5: Worst values $\pi^{\max \text{ latency}}$ for the optimal placements $\mathcal{P}_0^{\max \text{ latency}}$ (solid lines) and $\mathcal{P}_C^{\max \text{ latency}}$ (dashed lines) for different numbers of controller failures (0 failures: green, 2 failures: blue, and 4 failures: red) for the Topology Zoo topologies considering the optimal placements with $k = 5$ controllers.

Network Disruption Tolerance

In the following, the issue of network component failures and the resulting risk of a network disruption is addressed. The presented results are focusing on the node failure case. Link failures have been regarded as well but yield no new insights. In particular, node failures obviously lead to a higher risk of network disruption than link failures as naturally any node failure also involves the outage of all adjacent links of the failed node. As shown in Section 4.1.2, a simultaneous outage of two nodes in the Internet2 OS³E topology can lead to up to eight controller-less nodes when the placement of $k = 5$ controllers is done only according to $\pi_0^{\max \text{ latency}}$. As explained before, a node is considered controller-less if it is still working and part of a working subtopology (consisting of at least one more node), but cannot reach any controller. Nodes that are still working, but cut off without any working neighbors, i.e., subtopologies consisting only of a single node, are not considered to be controller-less. This is a valid assumption, as during this failure

situation these nodes have nobody to communicate with. Furthermore, without this relaxation, each node having at most two neighbors would have to contain a controller to not be controller-less when both neighbors fail.

Before giving a formal definition of the metric $\pi_{\mathcal{N}}^{\text{controller-less}}$, first, the extension of $\pi_{\emptyset}^{\text{max latency}}$ to the node failure case is given. Analogously to the controller failure case \mathcal{C} , a set of different node failure scenarios $s \in \mathcal{N}$ is considered. As in case of node failures the shortest path distances between different nodes change, distance matrices $d_{v,w}^s$ are introduced containing the distances between nodes v and w for a given failure scenario s . In particular, $d_{v,w}^{\emptyset} = d_{v,w}$. Based on these variables, $\pi_{\mathcal{N}}^{\text{max latency}}$ is defined as follows:

$$\pi_{\mathcal{N}}^{\text{max latency}}(\mathcal{P}) = \max_{(s \in \mathcal{N})} \pi_s^{\text{max latency}}(\mathcal{P}) = \max_{(s \in \mathcal{N})} \max_{(v \in \mathcal{V})} \min_{(p \in \mathcal{P})} d_{v,p}^s. \quad (4.6)$$

To avoid a value of $\pi_{\mathcal{N}}^{\text{max latency}}(\mathcal{P}) = \infty$ in case of controller-less nodes if there are values $d_{v,w}^s = \infty$, $\pi_{\mathcal{N}}^{\text{max latency}}(\mathcal{P})$ is set to the largest latency value of all "non controller-less" nodes.

$\pi_{\mathcal{N}}^{\text{controller-less}}$ is defined as the maximum number of controller-less nodes appearing for a certain placement when considering all failure scenarios of \mathcal{N} . Based on the distance matrices $d_{v,w}^s$, disconnection matrix variables $e_{v,w}^s$ are introduced with $e_{v,w}^{s_i} = 1$ if and only if the corresponding entry in $d_{v,w}^{s_i} = \infty$, i.e. in a particular failure scenario s_i , node v cannot reach node w . All other entries of matrix $e_{v,w}^s$ are 0. Using these disconnection matrices, $\pi_{\mathcal{N}}^{\text{controller-less}}$ is defined as

$$\pi_{\mathcal{N}}^{\text{controller-less}}(\mathcal{P}) = \max_{(s \in \mathcal{S})} \sum_{(v \in \mathcal{V})} \min_{(p \in \mathcal{P})} e_{v,p}^s. \quad (4.7)$$

Given these assumptions, the lowest possible value $\pi_{\mathcal{N}}^{\text{controller-less}}$ that can be reached by a placement of $k = 5$ controllers in the Internet2 OS³E topology when considering up to two node failures is two. This is due to the fact that all

scenarios s of up to two node failures are reflected in $\pi_{\mathcal{N}}^{\text{controller-less}}$ and even for the best possible placements of $k = 5$ controllers, there still remain subtopologies of two nodes that are controller-less in particular double node failure cases. An example placement $\mathcal{P}_{\mathcal{N}}^{\text{controller-less}}$ leading to the value $\pi_{\mathcal{N}}^{\text{controller-less}} = 2$ is shown in Figure 4.6. The question is limited to two simultaneous failures for two reasons: First, more simultaneous failures are unlikely to happen. Second, if more than two arbitrary failures happen in the same time, the topology can be totally disrupted so that basically no controller placement would help here anymore.

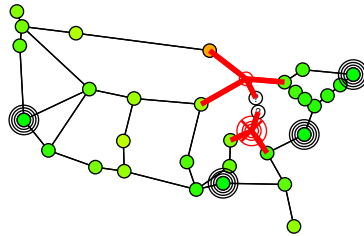


Figure 4.6: Illustration of an optimal placement $\mathcal{P}_{\mathcal{N}}^{\text{controller-less}}$ minimizing the number of controller-less nodes in at most two node-failures.

To illustrate $\pi_{\mathcal{N}}^{\text{controller-less}}$ for different controller placements, the topology is colored to indicate how often a certain node is controller-less when considering all different failure scenarios. A traffic-light scheme is used, where green depicts that a node is never controller-less in any scenario and red that it is controller-less in more than three failure scenarios. In between the values for green and red, a logarithmic scale is applied. Figure 4.7 shows the described color scheme for two placements $\mathcal{P}_{\emptyset}^{\text{max latency}}$ and the placement from Figure 4.6, $\mathcal{P}_{\mathcal{N}}^{\text{controller-less}}$.

In previous considerations, each node is considered to be of equal importance and $\pi_{\mathcal{N}}^{\text{controller-less}}$ just indicated the maximum number of simultaneously controller-less nodes. However, depending on the concrete application of SDN

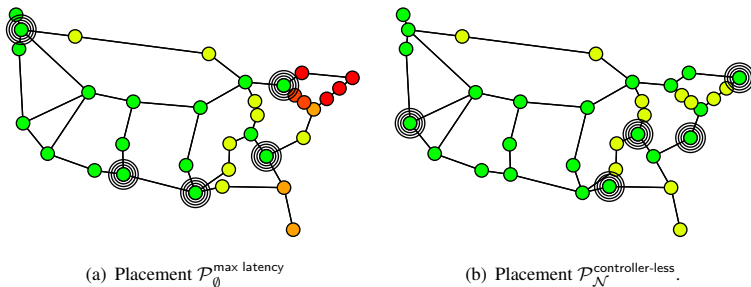


Figure 4.7: Illustration for different placements of how often a certain node is controller-less when considering all failure scenarios.

and centralized controllers, different nodes may account for a different amount of signaling with the controller or are of different importance. To include this, each node of the Internet2 OS³E example topology is assigned a weight according to the population of the city where the node is located. The population values are obtained using *Wolfram Alpha* [158]. Figure 4.8 shows the same illustrations as Figure 4.7 but with population sizes included in the optimization as importance of certain nodes. The resulting best placement $\mathcal{P}_N^{\text{controller-less}}$ differs from the one obtained when counting all nodes with uniform weight 1.

Obviously, $\pi^{\text{controller-less}}$ can be reduced with increasing number of controllers in a network. Figure 4.9 shows for the consideration of one and two failures how $\pi_N^{\text{controller-less}}$ decreases with increasing controller number k for the best placements \mathcal{P} according to $\pi_0^{\max \text{ latency}}$ and $\pi_N^{\text{controller-less}}$. It can be seen that in the Internet2 OS³E topology with a number of $k = 7$ controllers it is possible to eliminate all controller-less nodes in all one and two failure scenarios.

This raises the general question what the minimum number of controllers k and their placement are to eliminate the occurrence of controller-less nodes for up to two link and node failures. Each subtopology consisting of at least two nodes,

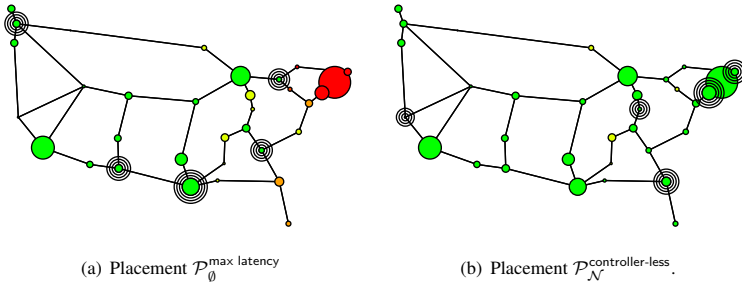


Figure 4.8: Illustration for different placements of how often a certain node is controller-less considering all failure scenarios including population sizes.

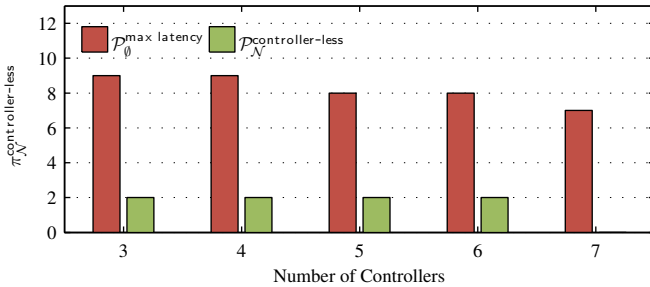


Figure 4.9: Controller-less nodes decrease with higher number of controllers k .

that can be cut off from the entire network by at most two link- or node-failures, has to be covered by at least one controller. Placing a controller in a subtopology also automatically covers all larger subtopologies that include the smaller one. This allows to develop a procedure to calculate k for a topology without evaluating $\pi_N^{\text{controller-less}}$ for all possible placements. Without this approach, the

computational effort to find an optimal k could not be handled. The minimum number of controllers can be found as follows. Firstly, find the set of all possible controller-less subtologies of at least two nodes that are not supertology of any other topology in the set. In Figure 4.10, there are eight such subtologies, so the *maximum* necessary number is $k = 8$. Secondly, find the minimum number of controllers necessary to cover all subtologies. In this case, in the right hand side of the topology, two controllers are enough to cover the three overlapping subtologies, thus $k = 7$ is enough².

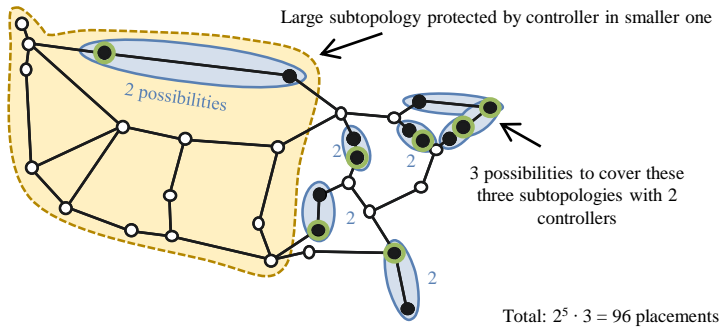


Figure 4.10: *Subtologies needing a controller to eliminate controller-less nodes.*

In this topology with $k = 7$, there are only 96 out of $\binom{34}{7} \approx 5.4$ million placements that are resilient, i.e. covering all subtologies. This shows that the protection against network disruption significantly reduces the fraction of possible controller placements, in this case to 0.002%. The nodes in Figure 4.10

²This procedure is similar to the procedure of finding a minimal set of positive boolean variables x_1, x_2, \dots, x_n to satisfy a boolean function given in "conjunctive normal form", e.g., $(x_1 \vee x_2 \vee x_3) \wedge (x_3 \vee x_5) \wedge \dots \wedge (x_{n-1} \vee x_n)$, where each literal x_i corresponds to one node $v_i \in \mathcal{V}$, each clause of \vee statements corresponds to one subtology and clauses being subsets of other clauses "absorb" these larger clauses.

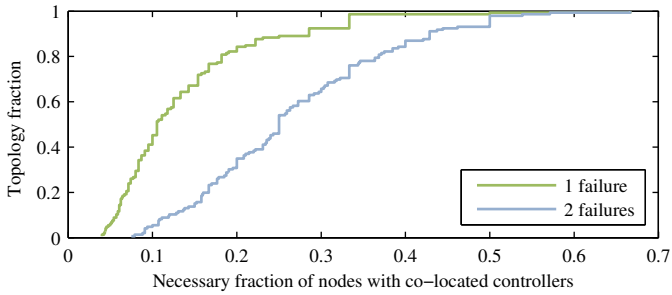
with a green border illustrate the best placement out of the 96 placements according to $\pi_{\emptyset}^{\max \text{ latency}}$. However, with $\pi_{\emptyset}^{\max \text{ latency}} = 44.9\%$, this placement is by far worse than the best possible placement with an equal number of nodes not fulfilling the resilience criterion ($\pi_{\emptyset}^{\max \text{ latency}} = 22.5\%$). That means that depending on how important $\pi_{\emptyset}^{\max \text{ latency}}$ is for an operator, much better results of this value could be obtained by paying the price of being not or not entirely resilient against controller-less nodes.

Before concluding this section, the investigations made for the Internet2 OS³E topology are extended to the Topology Zoo. In Figure 4.11, the minimum number of controllers (measured in percentage of nodes that require a controller) to eliminate controller-less nodes and the increase in $\pi^{\max \text{ latency}}$ to be resilient is illustrated for all 146 Topology Zoo topologies.

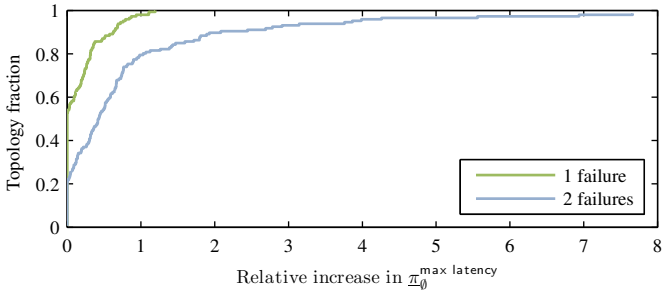
Obviously, the necessary fraction of nodes with co-located controllers is higher when resilience against two failures is required, compared to the one-failure-resilience. In 85% (125 out of 146) of the topologies it is enough to place $k = 2$ controllers to prevent controller-less subtopologies in any single node failure scenario. These are evidently all topologies that can not be split into subtopologies (larger than a single node) by a single node failure, i.e. at least two-connected (excluding potential single "appendix nodes") with a vertex connectivity larger than one. Depending on the network size, $k = 2$ results in a different fraction of nodes of a topology. When two arbitrary node failures should be covered, 50 out of 146 topologies need only $k = 3$ controllers, i.e. they are at least three-connected (excluding potential single "appendix nodes") with a vertex connectivity larger than two. All other topologies can be split into several subgraphs with two failures and need more controllers. The topologies that require a fraction of around 50% to be covered have a ring structure. To protect all subtopologies in a ring, every other node has to host a controller, leading to 50% fraction for an even node count or even more than 50% for an odd node count.

Finally, the price in terms of increased latencies that has to be paid to achieve resilience against controller-less nodes is investigated. Figure 4.11(b) shows the relative increase in the optimal $\pi_{\emptyset}^{\max \text{ latency}}$ between the best resilient placements

4 Resilience Enhancements for the Control Plane of SDN-based Core Networks



(a) Minimal necessary fraction of nodes with co-located controllers to fulfill the resilience property.



(b) Relative increase in $\pi_{\theta}^{\max \text{ latency}}$ to be resilient.

Figure 4.11: Evaluation of resilience against network disruption and controller-less nodes for different numbers of node failures and different topologies.

with minimal controller number k and the best placements with an equal number of controllers k that are not fulfilling the resilience criterion. The topologies where $\pi_{\theta}^{\max \text{ latency}}$ does not increase, i.e. is equally good as without resilience, mostly correspond to topologies, where the number k of necessary controllers is low. In this case, the fraction of resilient placements is high and in particular, often contains the best possible placements with regard to $\pi_{\theta}^{\max \text{ latency}}$. There are

however some topologies where the best values obtainable with a resilient placement are far worse than those obtainable without regarding the resilience requirements. An option to obtain both, a resilient placement and competitive $\pi^{\max \text{ latency}}$ values, could be to place more than the minimum number of controllers. In this case, first, a resilient placement is accomplished by placing this minimum number of controllers. Then, additional controllers can be added subsequently to decrease the maximum latency. However, the larger number of controllers increases the complexity.

4.2.2 Further Aspects of Resilient Placements

In this section, further aspects of resilient controller placements are discussed. First, the focus is on node-to-controller load balancing. Afterwards, inter-controller latency in the placement process is addressed.

Balancing Controller Load

Depending on the use case, it can be desirable to have roughly equal load on all controllers, so that no controller is overloaded while others have only little work to do. In the following, a good balance of the node-to-controller distribution is addressed. As formal metric, the balance of a placement or rather the imbalance, $\pi^{\text{imbalance}}$, is defined as the offset to a totally balanced distribution, i.e., the difference between the number of nodes assigned to the controller with the most nodes and the number of nodes assigned to the controller with the fewest nodes.

As mentioned before, it is assumed that each node is assigned to its closest controller according to the distance matrix $d_{v,w}$. These assignments allow to define assignment matrices n_p^s containing for each failure scenario s and controller p the number of nodes assigned to this controller. $\pi_\emptyset^{\text{imbalance}}$ and $\pi_{\mathcal{X}}^{\text{imbalance}}$ are defined subsequently as follows:

$$\pi_\emptyset^{\text{imbalance}}(\mathcal{P}) = \max_{(p \in \mathcal{P})} n_p^\emptyset - \min_{(p \in \mathcal{P})} n_p^\emptyset, \quad (4.8)$$

$$\pi_{\mathcal{X}}^{\text{imbalance}}(\mathcal{P}) = \max_{(s \in \mathcal{X})} \left(\max_{(p \in \mathcal{P})} n_p^s - \min_{(p \in \mathcal{P})} n_p^s \right). \quad (4.9)$$

As shown in Figure 4.2(c), the node-to-controller distribution can be highly imbalanced if not considered while choosing the controller placement. If taken into account, the imbalance can be drastically reduced. There are actually many placements with $k = 5$, leading to each controller having either 5 or 6 nodes assigned and thus $\pi_{\emptyset}^{\text{imbalance}} = 1$. However, this leads to an increase of the corresponding $\pi_{\emptyset}^{\text{max latency}}$ values. Figure 4.12 illustrates the trade-off between the metrics $\pi_{\emptyset}^{\text{max latency}}$ and $\pi_{\emptyset}^{\text{imbalance}}$ by displaying the entire solution space. Analogous to the previous illustration, the figure also shows the mean values, as well as the Pareto-optimal values. For better visibility, the axes limits of the graph have been adapted to display only the most important range of the placement solution space. To give an impression, the worst placements have values of $\pi_{\emptyset}^{\text{max latency}} > 85\%$ of the diameter and $\pi_{\emptyset}^{\text{imbalance}} \geq 25$.

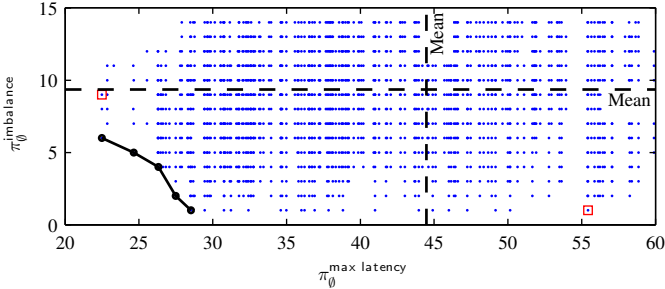


Figure 4.12: Trade-off between $\pi_{\emptyset}^{\text{max latency}}$ and $\pi_{\emptyset}^{\text{imbalance}}$.

The graph shows that there exist really well-balanced placements with optimal balance $\pi_{\emptyset}^{\text{imbalance}}$ that can have extremely bad $\pi_{\emptyset}^{\text{max latency}}$. The worst of these placements is illustrated by a red square in Figure 4.12. In this case, $\pi_{\emptyset}^{\text{max latency}}$ is even worse than the mean value of all possible placements. Similarly, placements

with lowest latency $\pi_{\emptyset}^{\max \text{ latency}}$ can have bad $\pi_{\emptyset}^{\text{imbalance}}$ values. The worst value is again marked with a red square. However, a good trade-off between both metrics is possible. The set of all Pareto-optimal values returned by POCO allows the network operator to choose one of the placements which seems to be the most adequate for their particular needs, e.g., a rather balanced one, or one with lower latencies.

To verify whether the observation also holds for other topologies, the trade-off between the optimal values for both parameters $\pi_{\emptyset}^{\max \text{ latency}}$ and $\pi_{\emptyset}^{\text{imbalance}}$ is evaluated for the topologies in the Topology Zoo. In about 20% of all topologies, there is one placement which is best according to both metrics. In the other 80% of topologies, the choice of the best placement according to one metric can significantly worsen the other. In these cases, the POCO-framework helps the operator to choose the most adequate and Pareto-optimal controller placement according to the network's policies.

In the context of controller-less nodes in Subsection 4.2.1, node weights based on city populations were already used to illustrate different importance of different nodes. For load balancing, and especially for some NFV use cases like fire-walling or monitoring, different nodes impose different load on the controllers. To illustrate this effect, Figure 4.13 depicts the best placements $\mathcal{P}_{\emptyset}^{\text{imbalance}}$ for two different node weights: i) uniform and ii) based on city populations. This consideration can also be extended from static weights to dynamic weights using POCO. E.g., the *Survivable fixed telecommunication Network Design Library (SNDlib)* [159] offers four topologies together with dynamic traffic matrices which can be imported in POCO and used to find the optimal controller placement considering a set of different node weights.

To complete the discussion on load imbalance, it is described how to combine both, the resilience requirements from this subsection and Subsection 4.2.1. In other words, a placement is looked for that offers a trade-off between load balancing and low maximum latency - not only in the failure-free case but also in case of controller failures. Altogether, this leads to four optimization objectives to be considered at the same time: $\pi_{\emptyset}^{\max \text{ latency}}$, $\pi_{\mathcal{C}}^{\max \text{ latency}}$, $\pi_{\emptyset}^{\text{imbalance}}$, and $\pi_{\mathcal{C}}^{\text{imbalance}}$.

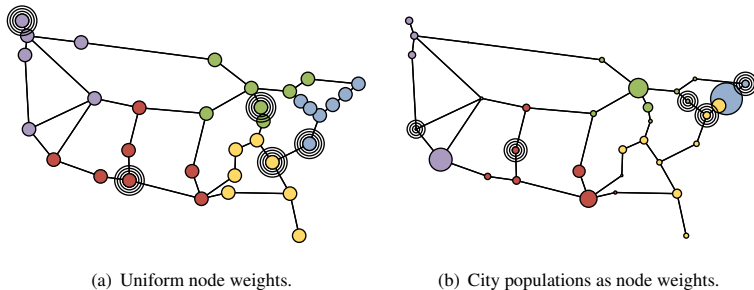


Figure 4.13: Placements $\mathcal{P}_\emptyset^{\text{imbalance}}$ for different node weights.

Knowing that usually the creation of a single objective function as (linear) combination of several objectives is not satisfying³, an intuitive approach to address this multi-criteria optimization would be introducing constraints to some of the metrics, e.g. $\pi_C^{\text{max latency}} < 80\%$ and $\pi_C^{\text{imbalance}} < 15$, and reducing the set of all placements to a smaller set of candidates fulfilling these constraints. Then, in a second step, the best placement out of this candidate set is chosen as adequate placement according to the remaining metrics $\pi_\emptyset^{\text{max latency}}$ and $\pi_\emptyset^{\text{imbalance}}$. However, this procedure can be unfavorable for two reasons. First, it demands a lot of knowledge about the network to choose limits for the constraints. Second, the risk is high to miss a much better result with respect to some metrics, if the result only slightly extends the limits given by the constraints. Therefore, another approach is used. The set of all Pareto-optimal values according to the four considered criteria is computed and shown in Figure 4.14.

To display the 4-dimensional solution space, the following illustration is chosen. Two out of the four dimensions, $\pi_C^{\text{max latency}}$ and $\pi_C^{\text{imbalance}}$, are chosen as x- and y-axis of the graph. The other two dimensions, $\pi_\emptyset^{\text{max latency}}$ and $\pi_\emptyset^{\text{imbalance}}$, are illustrated by different marker sizes and colors. For $\pi_\emptyset^{\text{max latency}}$, the traffic-light

³An example why this is the case is provided in the context of inter-controller latency at the end of Subsection 4.2.2

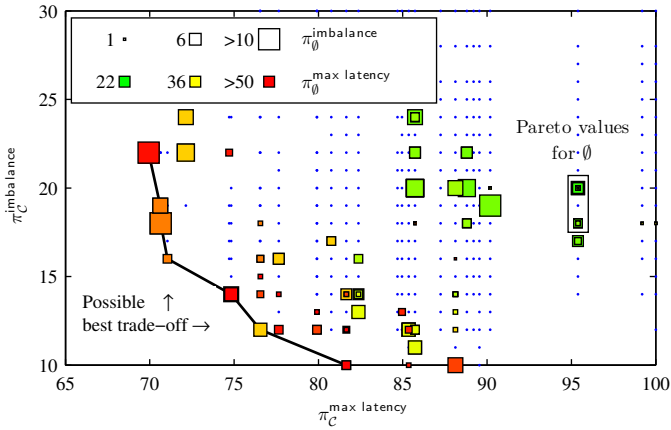


Figure 4.14: Trade-off between $\pi_{\emptyset}^{\max \text{ latency}}$, $\pi_C^{\max \text{ latency}}$, $\pi_{\emptyset}^{\text{imbalance}}$, and $\pi_C^{\text{imbalance}}$ displayed as 4-dimensional Pareto-optimal values.

color scheme as described before is used, and for $\pi_{\emptyset}^{\text{imbalance}}$ squares of different sizes, where a larger size indicates larger $\pi^{\text{imbalance}}$ values, i.e. worse balance. The points connected by black lines show the two-dimension Pareto-set according to $\pi_C^{\max \text{ latency}}$ and $\pi_C^{\text{imbalance}}$. The points in the background illustrate all the remaining solution space that is not part of the Pareto-optimal set. The area surrounded by a rectangle on the right side of the graph illustrates the location of the two-dimensional Pareto set according to $\pi_{\emptyset}^{\max \text{ latency}}$ and $\pi_{\emptyset}^{\text{imbalance}}$. Looking at this location shows that in particular, choosing one of the placements being Pareto-optimal for the failure-free case can result in arbitrarily bad performance when considering also the controller failure case. The constraint approach as described above could be illustrated in the figure, by looking only at these Pareto-values which are in the "lower left corner" of the graph and selecting the best, i.e. "greenest" and "smallest" square, out of those. Maybe the best trade-off, as indicated in

the figure, is reached, if the "smallest" or "lightest" square on the Pareto-optimal curve according to $\pi_C^{\max \text{ latency}}$ and $\pi_C^{\text{imbalance}}$ is chosen. In this case, the resilience is relatively well included in the trade-off, and still the failure-free case is on an acceptable level. All this information provided by the POCO-framework allows the network operators to choose the most adequate placement for their particular requirements. This can be either one of those described before or a totally different one.

Inter-Controller Latency

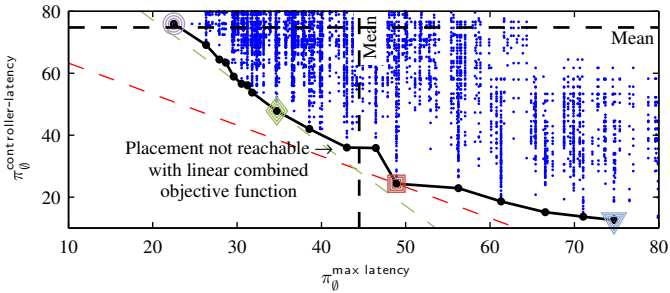
As last aspect of resilient controller placement, it is investigated how the inter-controller latency can be considered during the controller placement and what influence it has on the normal controller-to-node latency. Formally, the inter-controller latency $\pi^{\text{controller-latency}}$ is defined as the largest latency between any two controllers p_1, p_2 of a placement \mathcal{P} :

$$\pi^{\text{controller-latency}}(\mathcal{P}) = \max_{(p_1, p_2 \in \mathcal{P})} d_{p_1, p_2}. \quad (4.10)$$

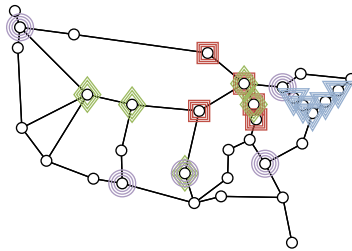
In [160], Levin et al. investigate the impact of a "logically centralized" but physically distributed network control plane, i.e., a control plane with nodes on different locations but with a central synchronized knowledge, on the operational performance of a network. However, they do not consider placement issues or the impact of failures on such an approach.

In general, without illustrating the placements here, all placements considering inter-controller latency tend to place all controllers much closer together. This increases the maximum latency from nodes to controllers.

Figure 4.15(a) shows the trade-off between both latency metrics for the failure-free case \emptyset . The Pareto-optimal values show that these two optimization goals cannot be achieved at the same time. To illustrate how the Pareto-optimal placements look like, Figure 4.15(b) shows four example placements out of the Pareto-optimal set: the best according to inter-controller latency, the best according to node-to-controller latency, the best when taking the average of inter-controller



(a) Solution space.



(b) Example placements.

Figure 4.15: Trade-off between node-to-controller and inter-controller latencies.

and node-to-controller latency, as well as the best when taking the weighted average with node-to-controller latency counting twice as much as inter-controller latency. It can be observed that when higher influence is given to node-to-controller latency, the controllers are more distributed in the network. Otherwise, when inter-controller-latency is given higher priority, the controllers are placed closer together. The graph also illustrates an example case, why the optimization of multiple criteria using an objective function as (linear) combination of those criteria does not reveal all Pareto-optimal solutions and might in particular not reveal those that might be preferable. E.g., the placement located at (43,36) would

not be obtained by the optimization according to any linear objective function $f = \alpha\pi_0^{\max\text{-latency}} + \beta\pi_0^{\text{controller-latency}}$, as for any choice of values α, β there are other placements (of the shown Pareto-optimal set) leading to better values of f .

4.2.3 POCO-Framework

To provide a simple and convenient way to calculate, illustrate and analyze the issues shown before, the already mentioned POCO-framework has been created. As said before, POCO is a Matlab-based framework to compute resilient Pareto-based Optimal COntroller-placements. By an efficient combined use of CPU and RAM, it can evaluate the entire solution space of all controller placements even when resilience is considered and thus provides information for all considered objectives for all placements. This offers the possibility to choose the best solution according to the particular requirements from the entire range of possible placements. Decisions on which objective of the multi criteria controller placement is most important can thus be postponed to after the optimization process.

In the following, POCO is briefly explained. First, the program structure and *Graphical User Interface (GUI)* is shown. Then, briefly the idea is summarized how the calculations can be significantly speedup using *MATLAB* matrix operations.

Program Structure

POCO has been implemented as a set of *MATLAB* scripts and functions allowing to calculate the different metrics considered in this work and to illustrate them. Furthermore, sets of Pareto-optimal values can be created according to multiple criteria. The realization as a set of scripts and functions allows for a good extendability of POCO. An overview on the program structure as well as on the contained functions is shown in Figure 4.16. The functions and scripts can be roughly classified into four groups: (1) input, (2) config, (3) calculation, and (4) output.

Graphical User Interface

To enhance the use of the POCO functions, a GUI has been implemented as well. The GUI provides easy access to all underlying functions and offers various choices how to illustrate the calculated results. A screenshot of the GUI can be seen in Figure 4.17. The upper part illustrates the topology according to different options. Amongst others, the illustration types displayed in Figure 4.2 can be selected. The lower part of the GUI allows to compare different placements and to easily access particular ones of them. This can be, e.g., the best according to a chosen metric, any arbitrary placement or one of the Pareto-optimal placements according to given metrics. In particular, the lower part also shows an evaluation of all placements according to a selection of two arbitrary metrics, as shown, e.g., in Figures 4.4 or 4.12, and highlights the Pareto-optimal results. All calculations necessary to evaluate different placements for different metrics and to offer an access to certain placements, are done only once – triggered by a menu item – and the results are then temporarily stored for quick access to speed up the illustration process.

Calculation Speedup using *MATLAB* Matrix Operations

The use of *MATLAB* allows for the fast computation of different controller placements and an efficient evaluation of the entire solution space for realistic network sizes. The idea how the calculations can be significantly speedup using *MATLAB* matrix operations is briefly explained in the following. Consider the example of the calculation of $\mathcal{P}_\emptyset^{\max \text{ latency}}$, i.e., the best placement according to $\pi_\emptyset^{\max \text{ latency}}$ as defined in Equation 4.1. Naively finding the best placement of k controllers involves for each placement: finding for each node the closest controller by calculating the latency to all controllers, possibly updating the maximum latency for that placement. Even if each single involved operation is simple, especially for large network instances, this is a time consuming process.

The calculation of the $\mathcal{P}_\emptyset^{\max \text{ latency}}$ value of a single controller placement can be significantly speedup using *MATLAB* matrix operations. The idea is exem-

4 Resilience Enhancements for the Control Plane of SDN-based Core Networks

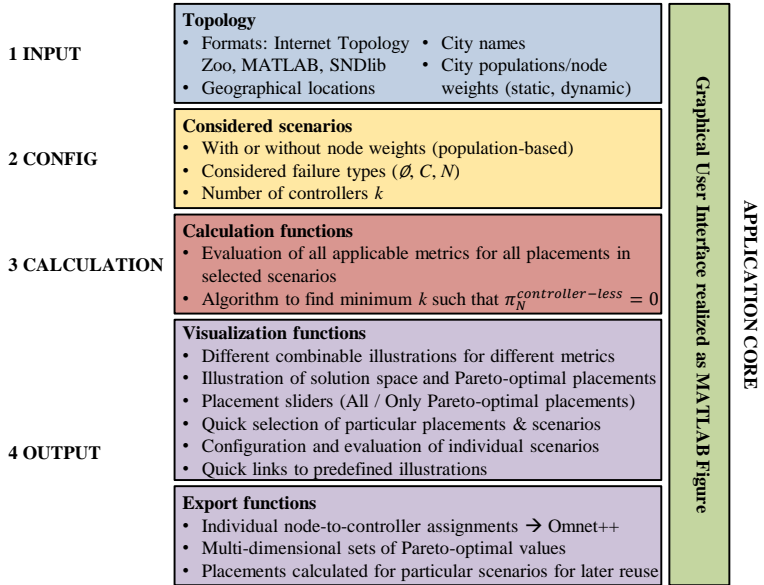


Figure 4.16: Program structure of POCO.

plarily illustrated for $k = 2$ controllers in Figure 4.18 and explained briefly in the following.

Let d be the shortest path distance matrix containing in each entry $d_{i,j}$ the distance from node i to node j . Hence, each column i of the matrix d corresponds to the distances of all different nodes to reach i . Let P be the matrix containing for each possible placement one row with the indexes of all controllers of this placement. Based on d and P , a 3-dimensional temporary matrix T is created containing for each placement in the 3rd dimension a 2-dimensional sub matrix with all columns of d corresponding to controller nodes of that particular placement. Using T , the placement $\mathcal{P}^{\max \text{ latency}}$ can be obtained simply by (1) building the row-wise minimum (best controller latency for each single node), (2) followed

4.2 Optimization of Resilient Controller Placement

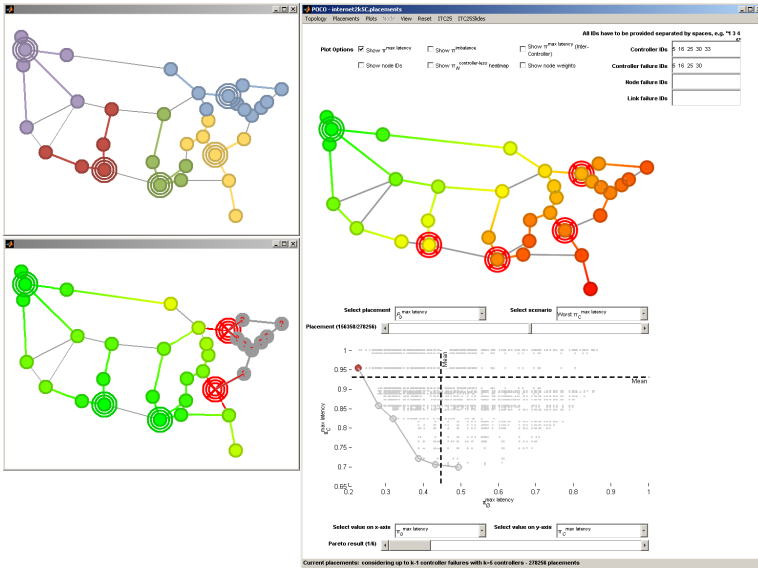


Figure 4.17: Screenshot of the POCO GUI.

by the column-wise maximum (π^{\max} latency for each placement), (3) followed by the minimum of the 3rd dimension ($\underline{\pi}^{\max}$ latency of all placements). Using this approach the entire calculation time even for large networks is usually in an order of seconds, as it is less a matter of available CPU power than rather of enough available RAM.

The other metrics considered in this chapter are calculated based on matrices similar to T . The metrics considering a number of different failure scenarios, e.g., $\pi_{\mathcal{N}}^{\max}$ latency, are not feasible to be calculated by the creation of a single matrix anymore as this matrix would increase exponentially with the number of considered failures. These metrics are therefore obtained by subsequently iterating the matrix calculations for each single failure scenario and aggregating the obtained results. Obviously, depending on the network size, these subsequent iterations

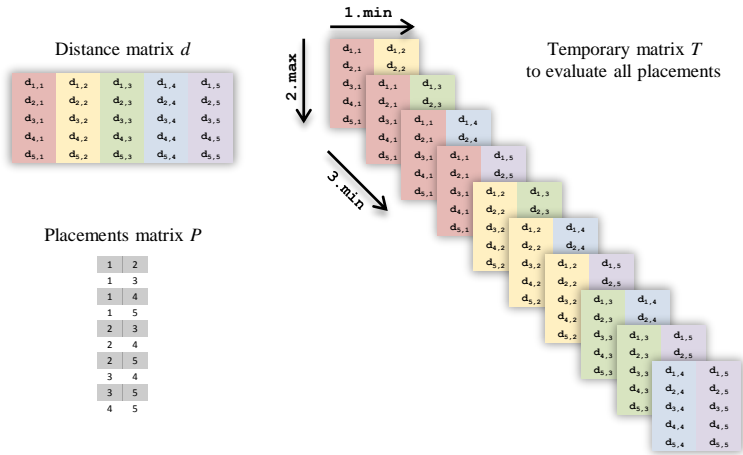


Figure 4.18: Visualization of the matrix based calculation for $k = 2$ controllers.

have the most significant impact on the run time of the entire calculation, because for topologies with n nodes, $\frac{n \cdot (n-1)}{2}$ different failure scenarios have to be calculated when considering even only up to two simultaneous node failures, or $\frac{m \cdot (m-1)}{2}$ scenarios for two link failures with m being the number of (undirected) links.

4.3 Related Work

Before concluding this chapter, an overview on related work will be given. Related work regarding resilience in general has been addressed in Chapter 3 and can be found, e.g., in [39, 161]. In this section, related work on the general controller placement problem and its variants relevant to the issues discussed in this chapter is provided.

As already mentioned and indicated by Heller et al. [147], the topic of general controller placement is well explored. In particular, the very basic version of con-

troller placement according to the latency of nodes to their controller, is also well discussed in the context of choosing the best location for plants, warehouses, or any other facilities in a given network topology. The problem is therefore also known as *plant, facility, or warehouse location problem* and it is a typical example for a MILP provided e.g. with the *IBM ILOG CPLEX* [148] software. If the objective is to minimize $\pi^{\max \text{ latency}}$, the problem is called *k-centers problem*, if the objective is $\pi^{\text{avg latency}}$, it is called *k-median* or *k-mean problem*. Further references to this general problem are provided in Heller's work [147]. Overviews on different aspects of the facility location problem and on different methodological approaches are also given in [162] in general and in [163] with the focus on "uncertainty" regarding e.g. uncertain traffic demands or latencies. These works however have a rather general and theoretical focus. They do not address the particular issues of controller placement in SDN networks with respect to multiple criteria and a focus on resilience. The following overview on related work focuses on variants of the controller placement problem which are closely related to the problem discussed here.

A variant of the problem similar to the node-to-controller balancing discussed here has been introduced by Archer et al. [164] as *load-balanced facility problem*. The objective is similar to $\pi^{\text{imbalance}}$. However, the authors address this problem in a different context concerning particular questions arising in the area of computer graphics. Furthermore, they provide only approximations to the problem regarding their particular optimization goals. In the context of load balancing, also the term *capacitated and uncapacitated facility problem* can be found, see e.g. [165] and contained references. The *capacitated* version assumes that the maximum number of nodes that can be assigned to a single controller is limited.

Different authors, among others Khuller et al. [166] and Chaudhuri et al. [167], look at variants called *fault tolerant* or *p-neighbor k-center problems*. These variants are similar to what is called "controller failure resilient placements" here. The works focus only on the theoretical methodology of the problem and provide approximation algorithms.

Apart of Heller et al. mentioned before [147], there are only few authors directly addressing facility location in the context of controller placement in

SDN networks. Bari et al. [168] address dynamic controller provisioning, i.e., controller placements changing over time depending on the current number of flows in the network. They propose an *Integer Linear Program (ILP)* formulation of their "Dynamic Controller Provisioning Problem" as well as two different heuristic algorithms to solve it for larger problem instances. The authors focus their metrics on flow setup time and minimal communication overhead regarding state synchronization. Controller or network failure issues or a combination of multiple criteria such as, e.g., $\pi^{\text{imbalance}}$ or $\pi^{\text{max latency}}$ are not addressed by their work. Zhang et al. [155] address a resilient optimization of the controller placement problem considering the outage of controllers or connections between nodes and controllers, i.e. network disruptions. They do not reassign nodes to new controllers if the original controller fails, but assume these nodes are controller-less. They propose a placement heuristic and simulation with the objective to minimize the number of such controller-less nodes. The very recent works of Hu et al. [169, 170] go in a similar direction. They introduce and compare different heuristic approaches to increase the resilience of a software defined network against connection failures between nodes and controllers. All these works [155, 169, 170] focus only on resilience against network failures and do not consider any additional metrics such as $\pi^{\text{imbalance}}$ or $\pi^{\text{max latency}}$. In particular, the trade-off between their metrics and other objectives, such as $\pi^{\text{max latency}}$, is not addressed. Furthermore, compared to the evaluation of the entire solution space, as it is done by POCO, no guarantee for the optimality of the presented results can be given.

Regarding the methodology of multi criteria/multi objective facility location, as already mentioned, in literature, there also exist various approaches for multi criteria facility location for a given combination of certain objectives, see e.g. [149–154] and references within. However, as stated before, the approach chosen here of evaluating the entire solution space offers – for realistic network sizes – the most freedom to consider different objectives as the decision on the most important metrics can be postponed to after the optimization.

4.4 Lessons Learned

In this chapter, resilience enhancements for the control plane of SDN-based core networks were discussed. In case of SDN networks, routing decisions are moved to centralized control units whereas normal switches are only forwarding traffic according to the rules provided by the controllers. When designing a centralized network control architecture, it is crucial to determine how many controllers are required and where they should be located in the network. In this chapter, these questions were addressed including different important aspects: quality in terms of maximum latencies between nodes and controllers as well as resilience in terms of failure tolerance and load balancing. Different resilience aspects were discussed and different metrics to describe these aspects were introduced. It was shown that the optimal values for the metrics quality and resilience are often impossible to achieve at the same time and adequate trade-offs have to be found. An evaluation on a large set of 146 topologies from the Topology Zoo underlines the validity of the presented findings.

First, the placement of controllers regarding the maximum latency not only in the failure-free case but also in case of controller failures was discussed. When controller failures appear, the nodes previously assigned to the failed controllers are reassigned to their nearest still working controller. This controller might however be possibly located at the other end of the network. The investigations showed that, if this is not considered during the placement selection process, the quality in terms of latencies between nodes and controllers can become arbitrarily bad in failure cases. As an example, it was shown that in most of the topologies, when placing $k = 5$ controllers the maximum latency between nodes and controllers in the failure-free case, $\pi_{\emptyset}^{\max \text{ latency}}$, can be kept below 30% of the network's diameter. However, considering the same placements if all except for one controller fail, in about 70% of the considered networks, the latency in the worst controller failure cases, $\pi_C^{\max \text{ latency}}$, increases to up to 100% of the diameter. The results showed that this issue can be regarded during the controller placement process but it is often not possible to optimize both metrics $\pi_{\emptyset}^{\max \text{ latency}}$

and $\pi_C^{\max \text{ latency}}$ at the same time. A consideration of Pareto-optimal placements offers network operators the possibility to choose the best trade-off for their particular needs.

The next issue concerning resilience that was discussed in this chapter is resilience against the outage of network elements, such as node failures. In particular, it was revealed that in most of the topologies more than 20% of all nodes need to be controllers to assure a continuous connection of all nodes to one of the controllers in any arbitrary double node failure scenario. Elsewise, there can be large parts of a topology that cannot reach their controller anymore in particular failure scenarios. Even more controllers might be necessary if not only resilience against network failures is desired but also certain latency values should be reached.

Another issue related to resilience that was presented in this chapter, was load balancing between the different controllers. If the number of node-to-controller requests in the network increases, so does the chance of additional delays due to queuing at the controller system. To be resilient against controller overload, the assignment of nodes to the different controllers should therefore be well-balanced. The metrics $\pi_\emptyset^{\text{imbalance}}$ and $\pi_C^{\text{imbalance}}$ were introduced to describe the imbalance of the node-to-controller assignment in the failure-free case as well as in the case of controller outages. Similar to the other considered metrics, it was revealed that optimal values for $\pi_\emptyset^{\text{imbalance}}$ and $\pi_C^{\max \text{ latency}}$ are in general impossible to achieve at the same time and adequate trade-offs have to be found. In particular, with the example of $\pi_\emptyset^{\max \text{ latency}}$, $\pi_C^{\max \text{ latency}}$, $\pi_\emptyset^{\text{imbalance}}$, and $\pi_C^{\text{imbalance}}$, a way of illustrating a four-dimensional set of Pareto-optimal values was proposed that allows network operators to choose their desired trade-offs between up to four simultaneous optimization goals.

For resilience against controller or network element failures as well as for reasons of load balancing, more than a single controller in the network is necessary. The existence of more than a single controller, however raises new questions such as the inter-controller latency. This issue was also briefly addressed. Again Pareto-optimal results were shown that revealed that the two optimization goals

node-to-controller latency and inter-controller latency cannot be achieved at the same time.

Finally, this chapter introduced the framework for resilient Pareto-based Optimal COntroller-placement (POCO). It offers network operators a range of options to select the placement that is most adequate for their particular needs, e.g., certain constraints on maximum latency, failure tolerance, or maximum number of nodes per controller. Based on an efficient combined use of CPU and RAM, the Matlab-based framework is able to evaluate the entire solution space of controller placements for realistic network sizes. This offers the possibility to choose the best placement according to one objective or a set of several objectives after the computation process without previously limiting the solution space by assigning any a priori importance to certain metrics. The POCO-toolset used to produce the results presented in this chapter is available online [171].

5 Conclusion and Outlook

Routing optimization is a key technology for *Traffic Engineering (TE)* in communication networks. Adjusting the path layout of flows influences the load distribution in a network. This can be utilized to fulfill given service requirements. To support the increasing number of real-time applications, routing has to be resilient in case of failures. Current used routing mechanisms differ in their control approach, in their robustness against failures, or in their reaction time on failures. Therefore, each of these technologies involves different resilience issues and different methods for optimization. This monograph focused on the analysis and optimization of resilient routing in core communication networks. Based on three example types, namely, decentralized IP routing as used in *Open Shortest Path First (OSPF)* or in the *Intermediate System to Intermediate System Protocol (IS-IS)*, *Multiprotocol Label Switching (MPLS)*-based routing, and typically centralized routing using *Software Defined Networking (SDN)*, various questions concerning these technologies have been investigated. A discussion of the different technology-specific issues in particular also establishes a better general understanding of the different routing approaches realized by these technologies. Thereby, this thesis can help to reach a better differentiation of the routing approaches and to better estimate their diverse potential for the use in future network architectures.

As it is known from other examples, technologies change fast over time and the de facto standard of today is probably no longer used in a few years. Also, the deployment of some of the technologies regarded in this work, such as *Not-via addresses (NotVia)* or *Ordered FIB Updates (OFIB)* is not guaranteed. However, the key concepts presented in this monograph and the methods used are not lim-

ited to any particular technology. They can also be applied or easily adapted to future technologies. A concrete example is given in the following repeating the main contributions of this work.

Considering recent developments, it is probable that future core communication networks will be based on SDN. In SDN networks, there is a separation between control plane and data plane. Simple switches are only forwarding packets according to flow table rules imposed by special controller units. Chapter 4 of this thesis directly addressed SDN networks. Different resilience issues regarding the control plane were discussed. Adequate placements of the controllers in the network need to be found to address these problems. It was shown that for realistic network instances, an exhaustive evaluation of all possible placements is computationally feasible and can be effectively implemented. The large advantage of an exhaustive evaluation is that it allows to compare placements according to different metrics without the need for any a priori choice of the most important criteria. This gives network operators the possibility to choose the most adequate placements for their particular needs. The *Pareto-based Optimal COntroller-placement (POCO)* tool has been implemented to provide a unified interface to the different visualizations, metrics, and the comparison of various placements options.

Even though the investigations and results of the other chapters of this thesis focused on IP routing and MPLS-based routing, they still offer relevant input for the configuration and use of SDN based core communication networks. The routing layout in SDN-based networks is in general very similar to the routing layout known from MPLS-based routing or from pure IP routing. Any explicit path routing or routing based on IP shortest paths can also be realized using SDN. Of course, SDN in addition also offers the possibility to extend the routing layout by setting up individual flow rules on demand. However, for reasons of complexity, it seems reasonable to setup the normal routing layout similar as known from IP and MPLS. In Chapter 2, path layouts based on IP shortest paths and available explicit paths layouts without any shortest paths constraints were compared. The results showed that explicit path layouts can offer a much better optimiza-

tion potential in terms of much lower maximum relative link loads especially if multiple paths are allowed. However, this comes at the cost of a much higher configuration effort regarding, e.g., the local and global number of forwarding entries in the different nodes. These results can also be applied to the path layout process in SDN-based networks. To find an adequate trade-off between both layout approaches, the normal primary path layout and local backup paths for the most probable failure scenarios could be based on explicit paths using the methods known from MPLS and *MPLS Fast Reroute (MPLS-FRR)*. Additionally, link costs could be adequately optimized and configured, to assure a better path layout also in case of more severe failures when a fallback to an underlying IP network or another decentralized routing is helpful. Resilience in the data plane of SDN in case of outages is normally realized by sending new forwarding rules to the nodes in the network leading to a new routing layout. However, if controllers cannot be reached anymore, no rerouting is possible. In this case, a fallback to a normal IP routing in an underlying layer or a similar distributed routing approach could help to assure the continuation of the communication. In Chapters 2 and 3 of this thesis, several extensions to the NetOpt heuristic link cost optimizer were proposed to allow for an optimization of IP routing including different routing phases such as failure-free state, reconverged state in case of failures, or *IP Fast Reroute (IP-FRR)* state. In Chapter 3, a resilience framework proposed by Menth et al. [39,40] was used and extended by additional illustrations to visualize based on an example analysis that decentralized routing significantly reduces the risk of overload in the network also for more severe failures even if only optimized for a small subset of failures. Thus, an optimized IP like routing is also promising in the context of SDN at least as a fallback solution in case of severe outages.

Some of the results of the studies conducted on *Unique Shortest Path (USP)* in Chapter 2 and on *Loop-Free Convergence (LFC)* in Chapter 3 cannot directly be mapped to the use case of SDN networks. They involve the routing optimization including IP-FRR mechanisms which as such are rather unlikely to be directly used in the context of SDN. However, some of the general observations made in course of the discussions on USP and LFC provide an important background

knowledge for SDN networks. In Chapter 2, it was shown that in case of ambiguous path layouts the maximum relative link load in a network can significantly differ from the values expected during the optimization if based on wrong *Tie-Breaker (TB)* assumptions. This can render routing optimization useless. This finding has to be kept in mind also when designing and optimizing the routing in SDN networks. All forwarding rules need to be clearly specified to avoid the co-existence of different forwarding possibilities for a single packet. If load balancing is targeted by splitting flows towards the same destination on several paths, the TB defining which packet is sent on which path has to be clearly defined. Otherwise, overload might appear on one of the paths, while the other paths have a lot of spare capacity.

The discussion on LFC and a temporary load increase during the OFIB phase presented in Chapter 3 also offers important input for the design and operation of SDN networks. During the recalculation of the routing, e.g., when a failure appears or is resolved, inconsistent information in the network might exist. If some routers already forward packets according to a newly calculated routing and others still have their old forwarding information, loops can occur. These loops do not only lead to the looping traffic being lost but also other traffic can be impaired by loops when the relative link load on the involved links increases. Even if loops are avoided, e.g., by an adequate ordering of the different updates using OFIB, a temporary load increase might appear on single links of the network. The problems of temporary forwarding loops or a temporary load increase are not limited to normal IP routing. They can equally appear in SDN networks, e.g., when some nodes request new forwarding rules, while others stick to their previous behavior, or in any other situation when forwarding table entries are updated asynchronously in different nodes of the network. Even if updates are managed by a central control instance and rolled out in the entire network, a completely synchronized update of all nodes is rather infeasible, e.g., due to delays in the distribution of control instructions. Approaches similar to OFIB could be adapted also for a use in SDN based networks to avoid temporary loops. In addition, the discussion presented in Chapter 3 showed that the heuristic routing optimization

can be extended to consider the OFIB phases such that, independently of any particular update order, a temporary load increase is avoided. This knowledge could be also transferred to the routing optimization for SDN networks.

Summing up, in this monograph different issues concerning resilient routing in core communication networks have been addressed. For each of the various questions, adequate methods and tools could be applied. As it has been exemplarily discussed in this conclusion for SDN-based core networks, the key findings and concepts presented in this monograph and the methods used are not limited to any particular technology but can also be applied or easily adapted to future technologies. In the same way, the tools created or extended in the course of this monograph, the heuristic optimizer NetOpt, the resilience analysis tool ResiLyzer, and the controller placement tool POCO are not limited to the scope presented here. On the one hand, they can be almost directly utilized for other use cases that are beyond the scope of this monograph. On the other hand, the implemented code and the designed ways of illustration can also be extended or reused for the creation of other tools and frameworks to address problems that might occur in future technologies.

To conclude this thesis, issues that are interesting to be studied in the future are mentioned. This thesis covered the placement of controllers as well as the routing layout in core communication networks separately. A first interesting question is to study the direct interrelation between routing layout and controller placement in SDN networks. The ResiLyzer tool presented in course of Chapter 3 as well as the underlying resilience analysis framework concentrates on the resilience analysis of single layer networks. An interesting question is to study the extension or adaption of the approach to consider multilayer networks, e.g., during the network design process. The POCO tool provides a unified interface to the comparison of different controller placements. The placement studies presented in this thesis focused on various resilience aspects. In context of SDN networks however not only resilience aspects are interesting. One interesting question is to extend the studies on controller placement towards further issues such as placements under dynamic load and latency circumstances.

List of Acronyms

CCDF	Complementary Cumulative Distribution Function
ECMP	Equal-Cost MultiPath
FIB	Forwarding Information Base
FRR	Fast Reroute
GUI	Graphical User Interface
HC	Hop-Count
IETF	Internet Engineering Task Force
ILP	Integer Linear Program
IP	Internet Protocol
IP-FRR	IP Fast Reroute
IS-IS	Intermediate System to Intermediate System Protocol [33]
ISP	Internet Service Provider
LDP	Label Distribution Protocol [36]
LFA	Loop-Free Alternate [43]
LFC	Loop-Free Convergence

List of Acronyms

LP	Linear Program
LSA	Link State Advertisement
LSP	Label Switched Path
LSR	Label Switching Router
MILP	Mixed Integer Linear Program
MP	Merge Point
MP_{ex}	Explicit Multi Path
MPLS	Multiprotocol Label Switching
MPLS-FRR	MPLS Fast Reroute
MRC	Multiple Routing Configurations [47]
MRT	Maximally Redundant Tree [45, 46]
NetOpt	A heuristic link cost optimizer
NFV	Network Functions Virtualization
NHOP	Next Hop
NNHOP	Next-Next-Hop
NotVia	Not-via addresses [34, 51]
NP	Complexity Class "Nondeterministic Polynomial Time"
OFIB	Ordered FIB Updates [41]
OS³E	Open Science, Scholarship and Services Exchange
OSPF	Open Shortest Path First [32]

PLR	Point of Local Repair
POCO	A framework for the computation of Pareto-based Optimal COntroller-placements [171]
RCP	Rich Client Plattform
RFC	Request For Comments
ResiLyzer	A tool for the resilience analysis in packet-switched communication networks [123]
rSPT	Reverse Shortest Path Tree
RSVP-TE	Resource Reservation Protocol - Traffic Engineering [37]
SDN	Software Defined Networking
SNDlib	Survivable fixed telecommunication Network Design Library
SP_{ex}	Explicit Single Path
SSP	Single Shortest Path
TB	Tie-Breaker
TE	Traffic Engineering
TM	Traffic Matrix
USP	Unique Shortest Path
USP_{IP}	IP-based USP

Nomenclature

General variables

\mathcal{G}	Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of nodes and links.
\mathcal{V}	Set of all nodes.
\mathcal{E}	Set of all link.

Chapter 2

Failure scenarios

\mathcal{X}	Considered failure case. Shown results in Chapter 2 focus on $\mathcal{X} \in \{\emptyset, \mathcal{L}\}$.
\emptyset	Failure-free case.
\mathcal{L}	Link failure case - in Chapter 2, all single link failures are considered.

Metrics and related variables

ψ	Objective function optimized by the heuristic.
$\rho_{\mathcal{X}}^{\max}$	Maximum relative link load of all links in all protected failure scenarios \mathcal{X} (Equ. 2.1).
$\phi_{\mathcal{X}}^{\text{weighted}}$	Objective function based on the penalty function ϕ .
ϕ	Penalty function introduced by Fortz et al. [55], illustrated in Figure 2.5.
$\rho(l)$	Relative load of a link $l \in \mathcal{E}$.
k	Administrative cost value $k \in [1 : k_{\max}]$.
k_{\max}	Allowed maximum link cost.
\mathbf{k}	Vector of link costs representing one link cost configuration.

Nomenclature

$\pi_{\mathcal{X}}^{\text{avg path length}}$	Average path lengths for considered failure scenarios \mathcal{X} (Equ. 2.2,2.3).
$P_{v,w}^s$	Paths between any nodes $v, w \in \mathcal{V}$ in a failure scenario $s \in \mathcal{X}$.
$LSP_{v,w}^{\text{primary}}$	Set of all primary paths between nodes $v, w \in \mathcal{V}$.
$LSP_{v,w,p,l}^{\text{backup}}$	Set of backup paths that protect the path p between nodes v and w for the outage of a link l on path p .
$LSP_{v,w}^{\text{overall}}$	Set of all paths for a demand between nodes v and w .
$LSP^{\text{pathtype},n}$	Set of all pathtype paths that pass through node n with $\text{pathtype} \in \{\text{primary}, \text{backup}, \text{overall}\}$.
$-LSP^{\text{pathtype},n}$	Set of all incoming pathtype paths of a node n with $\text{pathtype} \in \{\text{primary}, \text{backup}, \text{overall}\}$.
$+LSP^{\text{pathtype},n}$	Set of all outgoing pathtype paths of a node n with $\text{pathtype} \in \{\text{primary}, \text{backup}, \text{overall}\}$.

Chapter 3

Network scenarios

\mathcal{Z}	Network scenarios $\mathcal{Z} = (\mathcal{X}, \mathcal{H})$.
\mathcal{X}	Failure scenarios. Shown results in Chapter 3 focus on all scenarios with a probability of at least $p_{\min} = 10^{-15}$. This corresponds to the failure patterns $\emptyset, L, N, LL, LN, NN, LLL, LLN$.
\emptyset	Failure-free case.
\mathcal{L}	Link failure case.
\mathcal{N}	Node failure case.
\mathcal{H}	Traffic scenarios. For the results shown in Chapter 3 only a single traffic matrix is regarded as the results focus on different failure rather than traffic scenarios.
$p(z)$	Probability of a networking scenario calculated by $p(z) = p(s) \cdot p(h)$.

	Metrics and related variables
$R_r^x(l)$	Mapping function based on overload probabilities. $R_r^x(l) = P(\rho(l) > x)$, see Figure 3.5.
$R_q^y(l)$	Mapping function based on relative link load quantiles. $R_q^y(l) = \inf(x : P(\rho(l) \leq x) \geq y)$, see Figure 3.5.
$\rho_{\mathcal{X}}^{\max}$	Maximum relative link load of all links in all protected failure scenarios \mathcal{X} (Equ. 2.1).
$\rho_{\mathcal{L}, \text{FRR}}^{\max}$	Maximum relative link load during failure-free state, reconverged failure state, and NotVia state (Phases I,II,IV of the full failure cycle, see Figure 3.10).
$\rho_{\mathcal{L}, \text{OFIB}}^{\max}$	Maximum relative link load additionally considering the OFIB phases (Phase III and V, see Figure 3.10), i.e., all phases of the full failure cycle.
$\rho(l)$	Relative load of a link $l \in \mathcal{E}$.
	Variables related to the Loop-Free Convergence
$rSPT(B)$	Reverse shortest path tree of of router B .
$rSPT_A(B)$	Reverse shortest path tree regarding a link $A \rightarrow B$. It is the subtree of $rSPT(B)$ that is attached to the router A , see Figure 3.11.
$side(A)$	Side of a link failure, see Figure 3.11.
$rSPT(Z)$	Original rSPT to Z in the failure-free case, see Figure 3.17.
$rSPT^*(Z)$	New rSPT to Z in the failure-case, see Figure 3.17.

Chapter 4

	General variables
$d_{v,w}$	Matrix containing the shortest path distances between all nodes v and w of the set of all nodes \mathcal{V} .
\mathcal{P}	Placement of controllers, being a non-empty subset of the power set $2^{\mathcal{V}}$.

$d_{v,w}^s$	Distance matrices containing the distances between nodes v and w for a given failure scenario s . In particular, $d_{v,w}^0 = d_{v,w}$.
$e_{v,w}^s$	Disconnection matrix variables with $e_{v,w}^{s_i} = 1$ if and only if the corresponding entry in $d_{v,w}^{s_i} = \infty$, i.e. in a particular failure scenario s_i , node v cannot reach node w . All other entries of matrix $e_{v,w}^s$ are 0.
n_p^s	Assignment matrixes containing for each failure scenario s and controller p the number of nodes assigned to this controller.
	Failure scenarios
\mathcal{X}	Considered failure case. Shown results in Chapter 4 focus on $\mathcal{X} \in \{\emptyset, \mathcal{C}, \mathcal{N}\}$.
\emptyset	Failure-free case.
\mathcal{C}	Controller failure case - if not specified differently, in Chapter 4, the simultaneous outage of up to all except for one controller is considered.
\mathcal{N}	Node failure case - if not specified differently, in Chapter 4, the simultaneous outage of up to two nodes is considered.
	Metrics and related variables
$\pi_{\mathcal{X}}^{\max \text{ latency}}$	Maximum node-to-controller latency (Equ. 4.1, 4.2, 4.3, 4.5, 4.6).
$\pi_{\mathcal{X}}^{\text{imbalance}}$	Node-to-controller distribution imbalance (Equ. 4.8, 4.9).
$\pi_{\mathcal{N}}^{\text{controller-less}}$	Maximum number of controller-less nodes (Equ. 4.7).
$\pi_{\mathcal{X}}^{\text{controller-latency}}$	Maximum inter-controller latency (Equ. 4.10).
$\underline{\pi}_{\mathcal{X}}^{\text{metric}}$	Best value π^{metric} for given metric and controller number k (Equ. 4.4).
$\mathcal{P}_{\mathcal{X}}^{\text{metric}}$	Any placement leading to $\underline{\pi}_{\mathcal{X}}^{\text{metric}}$.

Bibliography and References

— Bibliography of the Author —

— Journal Articles —

- [1] D. Hock, M. Hartmann, M. Menth, M. Pióro, A. Tomaszewski, and C. Żukowski, “Comparison of IP-Based and Explicit Paths for One-to-One Fast Reroute in MPLS Networks,” *Telecommunication Systems*, vol. 52, no. 2, 2013.
- [2] D. Hock, M. Hartmann, C. Schwartz, and M. Menth, “ResiLyzer: Ein Werkzeug zur Analyse der Ausfallsicherheit in paketvermittelten Kommunikationsnetzen,” *Praxis der Informationsverarbeitung und Kommunikation (PIK)*, vol. 34, no. 3, 2011.
- [3] M. Hartmann, D. Hock, and M. Menth, “Routing Optimization for IP Networks Using Loop-Free Alternates,” *currently under submission*, 2014.
- [4] H.-T. Nguyen, N. P. Ngoc, T.-H. Truong, T. T. Ngoc, D. N. Minh, V. G. Nguyen, T.-H. Nguyen, T. N. Quynh, D. Hock, and C. Schwartz, “Modeling and Experimenting Combined Smart Sleep and Power Scaling Algorithms in Energy-aware Data Center Networks.” *Simulation Modelling Practice and Theory*, vol. 39, 2013.
- [5] F. Wamser, D. Hock, M. Seufert, B. Staehle, R. Pries, and P. Tran-Gia, “Using Buffered Playtime for QoE-Oriented Resource Management of YouTube Video Streaming,” *Transactions on Emerging Telecommunications Technologies*, vol. 24, no. 3, 2013.

- [6] D. Hock, F. Wamser, M. Seufert, R. Pries, and P. Tran-Gia, "OCEAN: Optimized Control Center for Experience Enhancements in Access Networks," *Praxis der Informationsverarbeitung und Kommunikation (PIK)*, vol. 36, no. 1, 2013.
- [7] C. Żukowski, A. Tomaszewski, M. Pióro, D. Hock, M. Hartmann, and M. Menth, "Compact Node-Link Formulations for the Optimal Single Path MPLS Fast Reroute Layout," *Advances in Electronics and Telecommunications*, vol. 2, no. 3, 2011.
- [8] T. Hoßfeld, F. Lehrieder, D. Hock, S. Oechsner, Z. Despotovic, W. Kellerer, and M. Michel, "Characterization of BitTorrent Swarms and their Distribution in the Internet," *Computer Networks*, vol. 55, no. 5, 2011.
- [9] R. Pries, D. Hock, and D. Staehle, "QoE based Bandwidth Management Supporting Real Time Flows in IEEE 802.11 Mesh Networks," *Praxis der Informationsverarbeitung und Kommunikation (PIK)*, vol. 32, no. 4, 2010.

— Conference Articles —

- [10] M. Hartmann, D. Hock, M. Menth, and C. Schwartz, "Objective Functions for Optimization of Resilient and Non-Resilient IP Routing," in *International Workshop on the Design of Reliable Communication Networks (DRCN)*, Washington, DC, USA, 2009.
- [11] D. Hock, M. Hartmann, M. Menth, and C. Schwartz, "Optimizing Unique Shortest Paths for Resilient Routing and Fast Reroute in IP-Based Networks," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Osaka, Japan, 2010.
- [12] M. Pióro, A. Tomaszewski, C. Żukowski, D. Hock, M. Hartmann, and M. Menth, "Optimized IP-Based vs. Explicit Paths for One-to-One Backup in MPLS Fast Reroute," in *International Telecommunication Network Strategy and Planning Symposium (Networks), Best Paper Award*, Warsaw, Poland, 2010.

- [13] D. Hock, M. Hartmann, C. Schwartz, and M. Menth, "Effectiveness of Link Cost Optimization for IP Rerouting and IP Fast Reroute," in *International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB & DFT)*, Essen, Germany, 2010.
- [14] D. Hock, M. Menth, M. Hartmann, C. Schwartz, and D. Stezenbach, "ResiLyzer: A Tool for Resilience Analysis in Packet-Switched Communication Networks," in *International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB & DFT)*, Essen, Germany, 2010.
- [15] D. Hock, M. Hartmann, T. Neubert, and M. Menth, "Loop-Free Convergence using Ordered FIB Updates: Analysis and Routing Optimization," in *International Workshop on the Design of Reliable Communication Networks (DRCN)*, Kraków, Poland, 2011.
- [16] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks," in *International Teletraffic Congress (ITC)*, Shanghai, China, 2013.
- [17] V. Burger, T. Hoßfeld, D. Garcia, M. Seufert, I. Scholtes, and D. Hock, "Resilience in Enterprise Social Networks," in *Workshop Sozioinformatik*, Koblenz, Germany, 2013.
- [18] D. Hock, V. Bernardo, T. Zinner, F. Wamser, K. A. Hummel, M. Curado, R. Pries, T. Braun, and P. Tran-Gia, "Evaluating the Trade-Off Between Energy-Efficiency and QoE in Wireless Mesh Networks," in *International Conference on Communications and Electronics (ICCE)*, Hué, Vietnam, 2012.
- [19] D. Schwerdel, D. Hock, D. Günther, B. Reuther, P. Müller, and P. Tran-Gia, "ToMaTo - A Network Experimentation Tool," in *International ICST*

Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom), Shanghai, China, 2011.

- [20] C. Żukowski, A. Tomaszewski, M. Pióro, D. Hock, M. Hartmann, and M. Menth, “Compact Node-Link Formulations for the Optimal Single Path MPLS Fast Reroute layout,” in *European Teletraffic Seminar (ETS)*, Poznan, Poland, 2011.
- [21] N. Bayer, D. Hock, A. Roos, M. Siebert, B. Xu, V. Rakocevic, and J. Habermann, “VoIP Performance in MeshBed - A Wireless Mesh Network Testbed,” in *IEEE Semiannual Vehicular Technology Conference (VTC) Spring, Poster Paper*, Marina Bay, Singapore, 2008.
- [22] T. Hoßfeld, D. Hock, P. Tran-Gia, K. Tutschku, and M. Fiedler, “Testing the IQX Hypothesis for Exponential Interdependency between QoS and QoE of Voice Codecs iLBC and G.711,” in *ITC Specialist Seminar*, Karlskrona, Sweden, 2008.
- [23] R. Pries, D. Hock, N. Bayer, M. Siebert, D. Staehle, V. Rakocevic, B. Xu, and P. Tran-Gia, “Dynamic Bandwidth Control in Wireless Mesh Networks: A Quality of Experience based Approach,” in *ITC Specialist Seminar*, Karlskrona, Sweden, 2008.
- [24] D. Hock, N. Bayer, R. Pries, M. Siebert, D. Staehle, V. Rakocevic, and B. Xu, “QoS Provisioning in WLAN Mesh Networks Using Dynamic Bandwidth Control,” in *European Wireless*, Prague, Czech Republic, 2008.

— Software Demonstrations —

- [25] D. Hock, M. Hartmann, C. Schwartz, and M. Menth, “ResiLyzer: Ein Werkzeug zur Analyse der Ausfallsicherheit in paketvermittelten Kommunikationsnetzen,” in *Final round of the KuVS Communication Software Award*, Kiel, Germany, 2011. [Online]. Available: <http://www3.informatik.uni-wuerzburg.de/resilyzer>

- [26] D. Hock, F. Wamser, M. Seufert, R. Pries, and P. Tran-Gia, "OCEAN: Optimized Control Center for Experience Enhancements in Access Networks," in *Conference on Networked Systems (NetSys)*, Stuttgart, Germany, 2013.
- [27] F. Wamser, D. Hock, M. Seufert, T. Zinner, and P. Tran-Gia, "Demonstrating the Benefit of Joint Application and Network Control Within a Wireless Access Network," in *IEEE INFOCOM*, Turin, Italy, 2013.
- [28] M. Hartmann, D. Hock, M. Höfling, T. Neubert, and M. Menth, "FIRMS - Demonstration of a Mapping System for Loc/ID Split Internet Routing in G-Lab," in *Würzburg Workshop on IP: Visions of Future Generation Networks (EuroView)*, Würzburg, Germany, 2010.
- [29] F. Wamser, D. Hock, M. Seufert, R. Pries, and P. Tran-Gia, "Performance Optimization in Access Networks Using a Combined Control Strategy," in *Würzburg Workshop on IP: Visions of Future Generation Networks (EuroView)*, Würzburg, Germany, 2012.

— General References —

- [30] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage, "California Fault Lines: Understanding the Causes and Impact of Network Failures," in *ACM SIGCOMM*, New Delhi, India, 2010.
- [31] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2012–2017," 2013.
- [32] J. Moy, "RFC2328: OSPF Version 2," IETF, 1998.
- [33] ISO, "ISO 10589: Intermediate System to Intermediate System Routing Exchange Protocol for Use in Conjunction with the Protocol for Providing the Connectionless-Mode Network Service," Internet Standard, 1992/2002.

- [34] S. Bryant, S. Previdi, and M. Shand, “RFC6981: A Framework for IP and MPLS Fast Reroute Using Not-Via Addresses,” IETF, 2013.
- [35] P. Pan, G. Swallow, and A. Atlas, “RFC4090: Fast Reroute Extensions to RSVP-TE for LSP Tunnels,” IETF, 2005.
- [36] L. A. (Ed.), I. M. (Ed.), and B. T. (Ed.), “RFC5036: LDP Specification,” IETF, 2007.
- [37] D. O. Awduche, L. Berger, D.-H. Gan, T. Li, V. Srinivasan, and G. Swallow, “RFC3209: RSVP-TE: Extensions to RSVP for LSP Tunnels,” IETF, 2001.
- [38] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling Innovation in Campus Networks,” *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 38, no. 2, 2008.
- [39] M. Menth, M. Duelli, R. Martin, and J. Milbrandt, “Resilience Analysis of Packet-Switched Communication Networks,” *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, 2009.
- [40] M. Menth, “Design and Performance Evaluation of Control Mechanisms for the Future Internet,” Professorial thesis, Chair of Distributed Systems, University of Würzburg, 2011.
- [41] M. Shand, S. Bryant, S. Previdi, C. Filsfils, P. Francois, and O. Bonaventure, “RFC6976: Framework for Loop-Free Convergence Using the Ordered Forwarding Information Base (oFIB) Approach,” IETF, 2013.
- [42] M. Hartmann, “Optimization and Design of Network Architectures for Future Internet Routing,” PhD thesis, Chair of Distributed Systems, University of Würzburg, to be published in 2014.
- [43] A. Atlas and A. Zinin, “RFC5286: Basic Specification for IP Fast Reroute: Loop-Free Alternates,” IETF, 2008.

- [44] S. Bryant, C. Filtsils, S. Previdi, M. Shand, and N. So, “Remote LFA FRR,” <http://tools.ietf.org/html/draft-ietf-rtgwg-remote-lfa>, IETF RTGWG, 2014.
- [45] A. Atlas, R. Kebler, C. Bowers, G. Enyedi, A. Csaszar, J. Tantsura, M. Konstantynowicz, and R. White, “An Architecture for IP/LDP Fast-Reroute Using Maximally Redundant Trees,” <http://tools.ietf.org/html/draft-atlas-rtgwg-mrt-frr-architecture>, IETF RTGWG, 2014.
- [46] G. Enyedi, A. Csaszar, A. Atlas, C. Bowers, and A. Gopalan, “Algorithms for Computing Maximally Redundant Trees for IP/LDP Fast-Reroute,” <http://tools.ietf.org/html/draft-enyedi-rtgwg-mrt-frr-algorithm>, IETF RTGWG, 2013.
- [47] M. Menth and R. Martin, “Network Resilience through Multi-Topology Routing,” in *International Workshop on the Design of Reliable Communication Networks (DRCN)*, Island of Ischia (Naples), Italy, 2005.
- [48] M. Menth, M. Hartmann, R. Martin, T. Cicic, and A. Kvalbein, “Loop-Free Alternates and Not-Via Addresses: A Proper Combination for IP Fast Reroute?” *Computer Networks*, vol. 54, no. 8, 2010.
- [49] M. Menth and W. Braun, “Performance Comparison of Not-Via Addresses and Maximally Redundant Trees (MRTs),” in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ghent, Belgium, 2013.
- [50] A. Raj and O. Ibe, “A Survey of IP and Multiprotocol Label Switching Fast Reroute Schemes,” *Computer Networks*, vol. 51, no. 8, 2007.
- [51] M. Shand and S. Bryant, “RFC5714: IP Fast Reroute Framework,” IETF, 2010.

- [52] R. Martin, M. Menth, and K. Canbolat, "Capacity Requirements for the One-to-One Backup Option in MPLS Fast Reroute," in *IEEE International Conference on Broadband Communication, Networks, and Systems (BROADNETS)*, San Jose, CA, USA, 2006.
- [53] M. Menth, R. Martin, M. Hartmann, and U. Spoerlein, "Efficiency of Routing and Resilience Mechanisms in Packet-Switched Communication Networks," *European Transactions on Telecommunications (ETT)*, vol. 21, no. 2, 2010.
- [54] M. Menth, M. Hartmann, and R. Martin, "Robust IP Link Costs for Multilayer Resilience," in *IFIP-TC6 Networking Conference (Networking)*, Atlanta, GA, USA, 2007.
- [55] B. Fortz and M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," in *IEEE INFOCOM*, Tel-Aviv, Israel, 2000.
- [56] R. Martin, "Resilience, Provisioning, and Control for the Network of the Future," PhD thesis 02/2008, Chair of Distributed Systems, University of Würzburg, 2008.
- [57] R. Martin, M. Menth, and M. Hemmkepler, "Accuracy and Dynamics of Multi-Stage Load Balancing for Multipath Internet Routing," in *IEEE International Conference on Communications (ICC)*, Glasgow, Scotland, UK, 2007.
- [58] M. Thorup and M. Roughan, "Avoiding Ties in Shortest Path First Routing," AT & T, Shannon Laboratory, Florham Park, NJ, Technical Report, 2001.
- [59] M. Menth, F. Lehrieder, B. Briscoe, P. Eardley, T. Moncaster, J. Babiarz, A. Charny, X. J. Zhang, T. Taylor, K.-H. Chan, D. Satoh, R. Geib, and G. Karagiannis, "A Survey of PCN-Based Admission Control and Flow Termination," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 3, 2010.

- [60] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel," in *ACM SIGCOMM*, Pittsburgh, PA, USA, 2002.
- [61] P. Batchelor, B. Daino, P. Heinzmann, C. Weinert, J. Späth, B. Van Caenegem, D. Hjelme, R. Inkret, H. Jäger, M. Joindot, A. Kuchar, E. Le Coquil, G. M. F. M. B. Leuthold, P. and de Marchis, H.-P. Nolting, F. Tillerot, and N. Wauters, *Ultra High Capacity Optical Transmission Networks. Final Report of Action COST 239*. Zagreb, Croatia: Faculty of Electrical Engineering and Computing, University of Zagreb, 1999.
- [62] "The GEANT website," <http://www.geant.net/>, 2008.
- [63] A. Nucci, A. Sridharan, and N. Taft, "The Problem of Synthetically Generating IP Traffic Matrices: Initial Recommendations," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 35, no. 3, 2005.
- [64] M. Roughan, "Simplifying the Synthesis of Internet Traffic Matrices," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 35, no. 5, 2005.
- [65] D. Lucraft, A. Eremin, and F. Ajili, "Enforcing Path Uniqueness in Internet Routing," in *ACM Symposium on Applied Computing (SAC)*, Dijon, France, 2006.
- [66] M. Menth, "Efficient Admission Control and Routing in Resilient Communication Networks," PhD thesis 02/2004, Chair of Distributed Systems, University of Würzburg, 2004.
- [67] E. Mulyana and U. Killat, "An Alternative Genetic Algorithm to Optimize OSPF Weights," in *ITC Specialist Seminar*, Würzburg, Germany, 2002.
- [68] A. Riedl, "A Hybrid Genetic Algorithm for Routing Optimization in IP Networks Utilizing Bandwidth and Delay Metrics," in *IEEE Workshop on IP Operations and Management (IPOM)*, Dallas, TX, USA, 2002.

- [69] C. Reichert and T. Magedanz, "A Fast Heuristic for Genetic Algorithms in Link Weight Optimization," in *International Workshop on Quality of future Internet Services (QofIS)*, Barcelona, Spain, 2004.
- [70] M. P. Petterson, R. Szymanek, and K. Kuchcinski, "A CP-LP Hybrid Method for Unique Shortest Path Routing Optimization," in *International Network Optimization Conference (INOC)*, Spa, Belgium, 2007.
- [71] A. Riedl and D. A. Schupke, "Routing Optimization in IP Networks Utilizing Additive and Concave Link Metrics," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, 2007.
- [72] D. Xu, M. Chiang, and J. Rexford, "Link-State Routing with Hop-by-Hop Forwarding can Achieve Optimal Traffic Engineering," in *IEEE INFOCOM*, Phoenix, AZ, USA, 2008.
- [73] M. Ericsson, M. Resende, and P. Pardalos, "A Genetic Algorithm for the Weight Setting Problem in OSPF Routing," *Journal of Combinatorial Optimization*, vol. 6, no. 3, 2002.
- [74] S. Balon and G. Leduc, "Combined Intra- and Inter-domain Traffic Engineering using Hot-Potato Aware Link Weights Optimization," in *ACM SIGMETRICS, Short Paper*, Annapolis, MD, USA, 2008.
- [75] S. Srivastava, G. Agrawal, and D. Medhi, "Dual-Based Link Weight Determination Towards Single Shortest Path Solutions for OSPF Networks," in *International Teletraffic Congress (ITC)*, Beijing, China, 2005.
- [76] B. Fortz, J. Rexford, and M. Thorup, "Traffic Engineering with Traditional IP Routing Protocols," *IEEE Communications Magazine*, vol. 40, no. 10, 2002.
- [77] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, 2002.

- [78] ———, “Robust Optimization of OSPF/IS-IS Weights,” in *International Network Optimization Conference (INOC)*, Paris, France, 2003.
- [79] D. Yuan, “A Bi-Criteria Optimization Approach for Robust OSPF Routing,” in *IEEE Workshop on IP Operations and Management (IPOM)*, Kansas City, MO, USA, 2003.
- [80] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot, “IGP Link Weight Assignment for Transient Link Failures,” in *International Teletraffic Congress (ITC)*, Berlin, Germany, 2003.
- [81] A. Sridharan and R. Guerin, “Making IGP Routing Robust to Link Failures,” in *IFIP-TC6 Networking Conference (Networking)*, Ontario, Canada, 2005.
- [82] M. Menth and M. Hartmann, “Threshold Configuration and Routing Optimization for PCN-Based Resilient Admission Control,” *Computer Networks*, vol. 53, no. 11, 2009.
- [83] M. Pióro, Á. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek, and S. Kozdrowski, “On Open Shortest Path First Related Network Optimization Problems,” *Performance Evaluation*, vol. 48, no. 1-4, 2002.
- [84] B. Fortz and M. Thorup, “Increasing Internet Capacity Using Local Search,” *Computational Optimization and Applications*, vol. 29, no. 1, 2004.
- [85] M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2004.
- [86] S. Köhler, D. Staehle, and U. Kohlhaas, “Optimization of IP Routing by Link Cost Specification,” in *ITC Specialist Seminar*, Würzburg, Germany, 2002.

- [87] A. Sridharany, R. Guérin, and C. Diot, "Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks," in *IEEE INFOCOM*, San Francisco, CA, 2003.
- [88] C. Lopes and A. de Sousa, "Heuristics for the MPLS Network Design with Single Path Minimum Weight Routing," in *International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HETNET)*, Ilkley, UK, 2005.
- [89] Y. Wang, Z. Wang, and L. Zhang, "Internet Traffic Engineering without Full Mesh Overlaying," in *IEEE INFOCOM*, Anchorage, AK, USA, 2001.
- [90] K. G. Ramakrishnan and M. A. Rodrigues, "Optimal Routing in Shortest-Path Data Networks," *AT&T Bell Laboratories Technical Journal*, vol. 6, no. 1, 2002.
- [91] A. Tomaszewski, M. Pióro, M. Dzida, and M. Zagozdzon, "Optimization of Administrative Weights in IP Networks Using the Branch-and-Cut Approach," in *International Network Optimization Conference (INOC)*, Lisbon, Portugal, 2005.
- [92] A. Tomaszewski, M. Pióro, M. Dzida, M. Mycek, and M. Zagozdzon, "Valid Inequalities for a Shortest-Path Routing Optimization Problem," in *International Network Optimization Conference (INOC)*, Spa, Belgium, 2007.
- [93] S. Köhler, "Interior Gateway Routing Optimization and Quality of Service - Algorithms and Performance Study," PhD thesis 01/2006, Chair of Distributed Systems, University of Würzburg, 2006.
- [94] E. Mulyana and U. Killat, "A Hybrid Genetic Algorithm Approach for OSPF Weight Setting Problem," in *Polish-German Teletraffic Symposium (PGTS)*, Gdansk, Poland, 2002.

- [95] A. Riedl, "Optimized Routing Adaptation in IP Networks Utilizing OSPF and MPLS," in *IEEE International Conference on Communications (ICC)*, Anchorage, AK, USA, 2003.
- [96] N. Wang, K.-H. Ho, , and G. Pavlou, "Adaptive Multi-Topology IGP Based Traffic Engineering with Near-Optimal Network Performance," in *IFIP-TC6 Networking Conference (Networking)*, Singapore, 2008.
- [97] J. Harmatos, "A Heuristic Algorithm for Solving the Static Weight Optimisation Problem in OSPF," in *IEEE Globecom*, San Antonio, TX, USA, 2001.
- [98] M. Dzida, M. Zagozdżon, and M. Pióro, "Optimization of Resilient IP Networks with Shortest Path Routing," in *International Workshop on the Design of Reliable Communication Networks (DRCN)*, La Rochelle, France, 2007.
- [99] Y. Zuo and J. Pitts, "Impact of Link Weight Ranges on OSPF Weight Solutions," in *International Conference on Communications and Networking in China*, Shanghai, China, 2007.
- [100] M. P. Pettersson, R. Szymanek, and K. Kuchcinski, "A CP-LP Approach to Network Management in OSPF Routing," in *ACM Symposium on Applied Computing (SAC)*, Seoul, Republic of Korea, 2007.
- [101] C. Zhang, "A Novel Mathematical Model for the Unique Shortest Path Routing Problem," ArXiv e-prints, Technical Report, No. 2008arXiv0807.0038Z, 2008. [Online]. Available: <http://arxiv.org/abs/0807.0038>
- [102] M. Dzida, M. Zagozdżon, M. Zotkiewicz, M. P. Pettersson, M. Pióro, M. Duelli, and M. Menth, "Three Methods for Optimizing Single-Shortest Path Routing," in *Conference on Next Generation Internet (NGI)*, Kraków, Poland, 2008.

- [103] A. Bley, "An Integer Programming Algorithm for Routing Optimization in IP Networks," in *Annual European Symposium on Algorithms (ESA)*, Karlsruhe, Germany, 2008.
- [104] —, "A Lagrangian Approach for Integrated Network Design and Routing in IP Networks," in *International Network Optimization Conference (INOC)*, Paris, France, 2003.
- [105] A. Li, P. Francois, and X. Yang, "On Improving the Efficiency and Manageability of NotVia," in *ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, New York, NY, USA, 2007.
- [106] R. Aggarwal, D. Papadimitriou, and S. Yasukawa, "RFC4875: Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)," IETF, 2007.
- [107] J. L. L. Roux, R. Aggarwal, J. Vasseur, and M. Vigoureux, "P2MP MPLS-TE Fast Reroute with P2MP Bypass Tunnels," <http://tools.ietf.org/html/draft-ietf-mpls-p2mp-te-bypass>, IETF, 2008.
- [108] A. Basu and J. G. Riecke, "Stability Issues in OSPF Routing," in *ACM SIGCOMM*, San Diego, CA, USA, 2001.
- [109] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving Sub-Second IGP Convergence in Large IP Networks," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 35, no. 2, 2005.
- [110] P. Francois and O. Bonaventure, "An Evaluation of IP-Based Fast Reroute Techniques," in *ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, Toulouse, France, 2005.
- [111] A. F. Hansen, T. Cicic, and S. Gjessing, "Alternative Schemes for Proactive IP Recovery," in *Conference on Next Generation Internet (NGI)*, Valencia, Spain, 2006.

- [112] M. Gjoka, V. Ram, and X. Yang, "Evaluation of IP Fast Reroute Proposals," in *IEEE International Conference on Communication System Software and Middleware (COMSWARE)*, Bangalore, India, 2007.
- [113] C. Filsfils, Ed., P. Francois, Ed., M. Shand, B. Decraene, J. Uttaro, N. Leyman, and M. Horneffer, "RFC6571: Loop-Free Alternate (LFA) Applicability in Service Provider (SP) Networks," IETF, 2012.
- [114] S. Litkowski, B. Decraene, C. Filsfils, K. Raza, M. Horneffer, and P. Sarkar, "Operational Management of Loop Free Alternates," <http://tools.ietf.org/html/draft-ietf-rtgwg-lfa-manageability>, IETF RTGWG, 2014.
- [115] H. Trong Viet, P. Francois, Y. Deville, and O. Bonaventure, "Implementation of a Traffic Engineering Technique that Preserves IP Fast Reroute in COMET," in *Rencontres Francophones sur les Aspects Algorithmiques des Telecommunications (ALGOTEL)*, Carry-Le-Rouet, France, 2009.
- [116] G. Retvari, J. Tapolcai, G. Enyedi, and A. Csaszar, "IP Fast ReRoute: Loop Free Alternates Revisited," in *IEEE INFOCOM*, Shanghai, China, 2011.
- [117] G. Retvari, L. Csikor, J. Tapolcai, G. Enyedi, and A. Csaszar, "Optimizing IGP Link Costs for Improving IP-level Resilience," in *International Workshop on the Design of Reliable Communication Networks (DRCN)*, Kraków, Poland, 2011.
- [118] L. Csikor, J. Tapolcai, and G. Retvari, "Optimizing IGP Link Costs for Improving IP-level Resilience with Loop-Free Alternates," *Computer Communications*, vol. 36, no. 6, 2013.
- [119] L. Csikor and G. Retvari, "IP Fast Reroute with Remote Loop-Free Alternates: The Unit Link Cost Case," in *International Congress on Ultra Modern Telecommunications and Control Systems (ICUMT)*, St. Petersburg, Russia, 2012.

- [120] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP Topologies with Rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, 2004.
- [121] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A Quantitative Comparison of Graph-Based Models for Internet Topology," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, 1997.
- [122] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic Matrix Estimation: Existing Techniques and New Directions," in *ACM SIGCOMM*, Pittsburgh, USA, 2002.
- [123] "ResiLyzer: A Tool for Resilience Analysis in Packet-Switched Communication Networks," <http://www3.informatik.uni-wuerzburg.de/resilyzer>.
- [124] P. Francois and O. Bonaventure, "Avoiding Transient Loops during the Convergence of Link-State Routing Protocols," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, 2007.
- [125] P. Cholda, A. Mykkeltveit, B. E. Helvik, O. J. Wittner, and A. Jajszczyk, "A Survey of Resilience Differentiation Frameworks in Communication Networks," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 4, 2007.
- [126] M. Duelli, "Heuristic Design and Provisioning of Resilient Multi-Layer Networks," PhD thesis 01/2012, Chair of Distributed Systems, University of Würzburg, 2012.
- [127] J. P. Sterbenz, D. Hutchinson, E. K. Cetinkaya, A. Jabbar, J. P. Rohrer, M. Schöllert, and P. Smith, "Resilience and Survivability in Communication Networks: Strategies, Principles, and Survey Disciplines," *Computer Networks*, vol. 54, no. 8, 2010.
- [128] P. Demeester, M. Gryseels, A. Autenrieth, C. Brianza, L. Castagna, G. Signorelli, R. Clemente, M. Ravera, A. Jajszczyk, D. Janukowicz, K. V.

- Doorselaere, and Y. Harada, "Resilience in Multilayer Networks," *IEEE Communications Magazine*, vol. 37, no. 8, 1999.
- [129] D. Medhi and D. Tipper, "Multi-layered network survivability-models, analysis, architecture, framework and implementation: An overview," in *DARPA Information Survivability Conference and Exposition (DISCEX)*, Hilton Head, SC, USA, 2000.
- [130] F. Rosenbaum, S. Jha, P. Boustead, and F. Safaei, "Resilience-differentiation in programmable virtual networks," in *IEEE International Conference on Communications (ICC)*, Paris, France, 2004.
- [131] I. B. Barla, D. A. Schupke, and G. Carle, "Analysis of resilience in virtual networks," in *Würzburg Workshop on IP: Visions of Future Generation Networks (EuroView)*, Würzburg, Germany, 2011.
- [132] ———, "Resilient virtual network design for end-to-end cloud services," in *IFIP-TC6 Networking Conference (Networking)*, Prague, Czech Republic, 2012.
- [133] J. F. Meyer, "Defining and evaluating resilience: A performability perspective," in *International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS)*, Eger, Hungary, 2009.
- [134] T. Panagiotis, "Measurement Frameworks and Metrics for Resilient Networks and Services: Challenges and Recommendations," ENISA European Network and Information Security Agency, Heraklion, 2010.
- [135] P. Van Mieghem, C. Doerr, H. Wang, J. M. Hernandez, D. Hutchison, M. Karaliopoulos, and R. Kooij, "A framework for computing topological network robustness," Delft University of Technology, Tech. Rep. 20101218, 2010.
- [136] M. Sveda, O. Rysavy, G. de Silva, P. Matousek, and J. Rab, "Reachability analysis in dynamically routed networks," in *IEEE International Confer-*

- ence and Workshops on Engineering of Computer Based Systems (ECBS), Las Vegas, NV, USA, 2011.
- [137] M. Goyal, M. Soperi, E. Baccelli, G. Choudhury, A. Shaikh, H. Hosseini, and K. Trivedi, "Improving convergence speed and scalability in ospf: A survey," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, 2012.
- [138] M. Shand and S. Bryant, "RFC5715: A Framework for Loop-Free Convergence," IETF, 2010.
- [139] A. Zinin, "Analysis and Minimization of Microloops in Link-state Routing Protocols," <http://tools.ietf.org/html/draft-ietf-rtgwg-microloop-analysis>, IETF RTG WG, 2005.
- [140] P. Francois, M. Shand, and O. Bonaventure, "Disruption-Free Topology Reconfiguration in OSPF Networks," in *IEEE INFOCOM*, Anchorage, AK, USA, 2007.
- [141] F. Clad, P. Merindol, J.-J. Pansiot, P. Francois, and O. Bonaventure, "Graceful Convergence in Link-State IP Networks: A Lightweight Algorithm Ensuring Minimal Operational Impact," *accepted for IEEE/ACM Transactions on Networking*, 2013.
- [142] T. Yoshihiro and M. Jibiki, "Single Node Protection Without Bouncing in IP Networks," in *IEEE Workshop on High Performance Switching and Routing (HPSR)*, Belgrade, Serbia, 2012.
- [143] P. Francois and O. Bonaventure, "Avoiding Transient Loops during IGP Convergence in IP Networks," in *IEEE INFOCOM*, Miami, Florida, 2005.
- [144] J. Fu, P. Sjödin, and G. Karlsson, "Loop-free updates of forwarding tables," *IEEE Transactions on Network and Service Management (IEEE TNSM)*, vol. 5, no. 1, 2008.
- [145] L. Shi, J. Fu, and X. Fu, "Loop-Free Forwarding Table Updates with Minimal Link Overflow," in *IEEE International Conference on Communications (ICC)*, Dresden, Germany, 2009.

- [146] A. Tootoonchian and Y. Ganjali, "HyperFlow: a Distributed Control Plane for OpenFlow," in *Internet Network Management Workshop/Workshop on Research on Enterprise Networking*, San Jose, CA, USA, 2010.
- [147] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, Helsinki, Finland, 2012.
- [148] "Cplex," <http://www.cplex.com/>, ILOG, Inc.
- [149] U. Bhattacharya, J. R. Rao, and R. N. Tiwari, "Fuzzy Multi-Criteria Facility Location Problem," *Fuzzy Sets and Systems*, vol. 51, no. 3, 1992.
- [150] M. Ehrgott, *Multicriteria Optimization*. Berlin, Germany: Springer, 2005.
- [151] I. Harris, C. Mumford, and M. Naim, "The Multi-Objective Uncapacitated Facility Location Problem for Green Logistics," in *IEEE Congress on Evolutionary Computation (CEC)*, Trondheim, Norway, 2009.
- [152] A. Lancinskas and J. Zilinskas, "Solution of Multi-Objective Competitive Facility Location Problems Using Parallel NSGA-II on Large Scale Computing Systems," in *Applied Parallel and Scientific Computing*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer, 2013, vol. 7782.
- [153] T. Xifeng, Z. Ji, and X. Peng, "A Multi-Objective Optimization Model for Sustainable Logistics Facility Location," *Transportation Research Part D: Transport and Environment*, vol. 22, 2013.
- [154] S. H. A. Rahmati, V. Hajipour, and S. T. A. Niaki, "A Soft-Computing Pareto-based Meta-Heuristic Algorithm for a Multi-Objective Multi-Server Facility Location Problem," *Applied Soft Computing*, vol. 13, no. 4, 2013.
- [155] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks," in *IEEE Globecom*, Houston, TX, USA, 2011.

- [156] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A Flexible OpenFlow-Controller Benchmark," in *European Workshop on Software Defined Networks (EWSDN)*, Darmstadt, Germany, 2012.
- [157] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, 2011.
- [158] "Wolfram Alpha," <http://www.wolframalpha.com>.
- [159] S. Orlowski, R. Wessály, M. Pióro, and A. Tomaszewski, "SNDlib 1.0 - Survivable Network Design Library," *Networks*, vol. 55, no. 3, 2009.
- [160] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically Centralized? State Distribution Trade-offs in Software Defined Networks," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, Helsinki, Finland, 2012.
- [161] M. Menth, R. Martin, and J. Charzinski, "Capacity Overprovisioning for Networks with Resilience Requirements," in *ACM SIGCOMM*, Pisa, Italy, 2006.
- [162] Z. Drezner, *Facility Location: A Survey of Applications and Methods*. Berlin, Germany: Springer, 1995.
- [163] S. H. Owen and M. S. Daskin, "Strategic Facility Location: A Review," *European Journal of Operational Research*, vol. 111, no. 3, 1998.
- [164] A. Archer and S. Krishnan, "Importance Sampling via Load-Balanced Facility Location," in *Conference on Integer Programming and Combinatorial Optimization (IPCO)*, Bertinoro, Italy, 2008.
- [165] F. J. F. Silva and D. S. de la Figuera, "A Capacitated Facility Location Problem with Constrained Backlogging Probabilities," *International Journal of Production Research*, vol. 45, no. 21, 2007.

- [166] S. Khuller, R. Pless, and Y. Sussmann, "Fault Tolerant k-Center Problems," in *Italian Conference on Algorithms and Complexity (CIAC)*, Rome, Italy, 1997.
- [167] S. Chaudhuri, N. Garg, and R. Ravi, "The p-Neighbor k-Center Problem," *Information Processing Letters*, vol. 65, no. 3, 1998.
- [168] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic Controller Provisioning in Software Defined Networks," in *International Conference on Network and Services Management (CNSM)*, Zürich, Switzerland, 2013.
- [169] Y. nan Hu, W. dong Wang, X. yang Gong, X. rong Que, and S. duan Cheng, "On the Placement of Controllers in Software-Defined Networks," *Journal of China Universities of Posts and Telecommunications*, vol. 19, no. 2, 2012.
- [170] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware Controller Placement for Software-Defined Networks," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ghent, Belgium, 2013.
- [171] "POCO: A Framework for the Computation of Pareto-based Optimal Controller-Placements," <http://www3.informatik.uni-wuerzburg.de/poco>.

ISSN 1432-8801