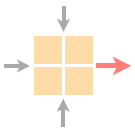
ETH zürich



Networked Systems
ETH Zürich — seit 2015

Reducing the Overhead of SRv6

Master Thesis

Author: Benjamin Arn

Tutors: Jean Mégret, Ricardo Santos, Dr. Tibor Schneider

Supervisor: Prof. Dr. Laurent Vanbever

January 2026 to July 2026

Abstract

Internet traffic is ever-growing, and network operators are in constant search for ways to use the networks to their full capacity. One approach is to use Traffic Engineering (TE) to spread traffic around the network to prevent congestions. Usually, this is done by minimizing the Maximal Link Utilization (MLU), i.e. the relative utilization of the bottleneck link. Segment Routing for IPv6 (SRv6) enables rerouting of default shortest path routes by appending IPv6 addresses to the SRv6 header, providing the fine-grained control needed for good TE.

But not all is well: the MLU does not tell the full story. It underestimates real traffic by ignoring the additional load introduced by the SRv6 header itself. For 2SR, one of the most popular Segment Routing TE algorithms, the MLU gets underestimated by 1.91% in the average network. Also, the MLU cannot accurately predict how resilient a solution is to random traffic variations.

We run a comprehensive comparison between existing SR-TE approaches to gain detailed insight into their differences. To account for the inaccurate calculation of the MLU, we adapt current SR-TE approaches and are able to lower the real MLU value of 2SR by 1.64% again. As an option to compare different configurations beyond the MLU value, we propose and discuss 3 different metrics to assess resilience to varying and increasing demands. We show with a sampling based experiment that the solution with the best MLU fares the worst under strong traffic variations, and that it is worth to minimize not only the MLU but also the number of rerouted paths.

Contents

1 Introduction **1**

2 Related Work **4**

3 Overview of Traffic Engineering with SRv6 **6**
3.1 Segment Routing with SRv6 6
3.1.1 IPv6 Routing 6
3.1.2 Segment Routing Principles 6
3.1.3 The SRv6 Header 7
3.2 Traffic Engineering 7
3.2.1 Maximum Link Utilization 7
3.2.2 2SR: the Linear Programming Approach 9
3.2.3 TLE: Adding a Second Optimization Step to 2SR 9
3.2.4 SRLS: Local Search Optimization 9
3.3 The REPETITA Framework 10

4 Main **11**
4.1 A Comprehensive Analysis of Existing Approaches 11
4.2 Enhancing the Existing Algorithms 13
4.2.1 Estimating the SRv6-Header Overhead 13
4.2.2 Analyzing the Benefit of the Extension 14
4.3 Extending the Analysis with an Emphasis on Resilience 16
4.3.1 Motivation: MLU is Limited 16
4.3.2 Different Approaches for Estimating Resilience 16
4.3.3 Relating Sampling Based Resilience back to Simple Metrics 18

5 Outlook **20**

References **22**

Chapter 1

Introduction

The internet grows and will not stop growing. To handle challenges introduced by changing demands due to new technology and the general rise in internet traffic, network operators seek out ways to enhance network capabilities. IP-based routing dominates the internet, but sometimes the simplicity of its destination-based shortest path routing leads to congestions and otherwise undesired network states. Nonetheless, configuring all possible paths of a network individually is complicated, especially for bigger networks. Segment Routing (SR) supports rerouting of traffic flows by adding detours to routing paths to spread traffic around when needed, but still leaning on shortest path routing everywhere else. Thus, SR enables full-grained control over the network without the complexity that complete source-based routing brings.

Advancements in SR has lead to SRv6, the implementation of segment routing on top of IPv6 routing. With MPLS, which was the main way of using SR before, traffic was routed based on a system of labels on top of the underlying routing system. In contrary, SRv6 removes that overlay by using normal IPv6 routing, a protocol that hardware is already optimized for. Additionally, it is more scalable due to the size of the address space and more customizable due to the possibility to define custom networking functions. Routing inside a SRv6-enabled network is done with so-called Segment Identifier (SID), a special kind of IPv6 address used for detours using SR. If traffic gets rerouted, the path is set by adding SIDs to the SID-list inside the SRv6-Header as defined in [1], with the IPv6 header as the outermost layer having the first detour as the destination address. Thus, only routers used as a detour have to inspect the SRv6 header, leading to minimal computational overhead.

Rerouting traffic can be useful for a number of reasons. For example, flows can get routed through a firewall node, or get routed around some central control node. If a node or a link fails, the network converges to a new forwarding state, and during this state of convergence, routing loops can occur. With Topology Independent Loop-Free Alternate (TI-LFA) Fast Reroute, SR can solve that problem too: the node, that detects the failure first, sets up a detour that forces other nodes to use the updated paths, even if they do not know the new network state. Though, one of the central benefits of SR is the ability to do traffic engineering (TE): Incorporating information about the traffic in the network to spread it as evenly as possible to remove bottlenecks.

The configuration of these paths is still complicated, unfortunately. The paths that include detours have to be configured statically in the routers these paths originate from. Finding optimal paths for traffic engineering is a hard problem, and is usually done based on a snapshot of all flows between

all nodes of a network called the traffic matrix. The quality of such a SR configuration is usually measured through the Maximum Link Utilization (MLU). The (relative) link utilization describes the traffic over a link divided by its capacity, and the MLU is the biggest utilization of all links of the network, i.e. the utilization of the bottleneck links. If the MLU value is over 1, we know that at least one link is overutilized, which leads to packet losses and congestion. Computational complexity can important as well, as it allows for faster recalculation if the traffic changes.

But regardless of the way these SR configurations are found, the configurations by themselves are not well understood. The MLU optimizing SR-TE approaches we use have to be reliable, as the configurations are too complicated to understand on a path-to-path basis, especially for networks with many nodes. For example, the optimal solution might lead to paths that use the same link twice [2]; It is not directly obvious why those routes can be optimal. With the help of REPETITA [3], a framework done specifically for experimentation and comparison of SR traffic engineering algorithms, we try to look at MLU based optimization to see where it lacks and where it could be improved.

It becomes clear quickly that current algorithms are very good at finding solutions with minimal MLU. For simple networks, the solution quality is almost equal to the optimum routing solution without the restraints that SRv6 has in the real world, like a limited length of the SID list. When comparing solvers using linear programming with solvers that use heuristics like the local search algorithm SRLS [4], the found SR configurations vary drastically. This is exciting, we found another optimizing problem! Nevertheless, the MLU falls short in capturing the differences and cannot help in making an informed decision.

One problem with MLU is that it is missing a small, but crucial part of the traffic: the segment routing header. Calculated realistically, the segment routing header with a single SID is about 2.76% of the size of the average traffic packet. An overhead of that size should not be ignored, especially if it could occur on the bottleneck link. Experiments show this overhead to be 1.91% on average for 2SR, measured in increase in MLU. It can be seen that the local search algorithm SRLS is less affected by this overhead than 2SR. This is most likely due to heuristics used in SRLS that route flows away from the bottleneck link instead of towards it, and thus having less detours with SRv6 header on the bottleneck link. This problem is easy to solve, as the algorithms can be adjusted to the more accurate flow calculations, and we show that they find better paths under the new circumstances.

Only minimizing the MLU value can lead to a lot of wasted resources if done carelessly. In the simplest, most popular SR-TE algorithm [5], a lot of unnecessary detours occur. Detours have no MLU cost everywhere else but on the bottleneck link, and thus are not prevented when optimizing for MLU. This waste of resources might not be a problem at first, but leads to less resilience against changes in network traffic, which will inevitably happen. We describe different ways of measuring resilience, namely the growth vector [6], an analytical congestion prediction and a sampling-based congestion prediction. Experiments done with the sampling-based congestion prediction show that the solver, that minimizes the number of configured SR-paths in a two-step optimization, clearly performs best under random traffic variations. At the same time, the experiments indicate that neither the number of configured SR-paths nor any other simple metric are sufficient for predicting the performance under random traffic variations.

First, we discuss interesting work that has gone in a different direction than we did in 2. Then, we give an overview of the theory, conventions and algorithms that are used and discussed in this thesis in 3. In our main part 4, we discuss certain non-optimal aspects of the MLU by looking at

different ways of analyzing network traffic. Finally, we discuss how this work could be expanded and which new questions arise in **??**.

Chapter 2

Related Work

Topology Independent Loop-Free Alternate (TI-LFA) as defined in [7] is a fast rerouting mechanism and the other big selling point of SR besides SR-TE. If the network topology changes unexpectedly, e.g. when a node fails, all routers have to recalculate their forwarding tables. Not all nodes might learn about the network change at the same time, and thus forwarding loops can occur. Fast rerouting enables routers to switch to a pre-calculated backup path that is guaranteed to not include forwarding loops, if such paths exist. If SR is used, nodes that notice the failure early can use the SRv6 Header to enforce the routing they prefer. With TI-LFA, backup paths are not only guaranteed to exist but are indeed optimal. While TI-LFA solves another problem than SR-TE, some overlap can be found: In [8], TI-LFA backup paths are analyzed on simple TE-metrics. In [9], backup paths are calculated with a second SR configuration in mind, that ensures that no link gets congested for any possible failure. In [10], the protection TI-LFA offers is expanded to not only include link failures but congestion events as well.

A benefit of SRv6 running on top of an IPv6 network is that not all routers inside an SRv6 domain have to have SR capabilities, but only those used as a detour. Of course, SR-TE can be used even if not all nodes are SR capable. [11] gives algorithms that can be used to determine which non-SRv6 node should be upgraded to minimize MLU. [12], [13] and other works discuss how to use SR to reroute traffic optimally while turning off as many links as possible to save energy during times of low traffic.

If a configuration contains a lot of reroutes, the deployment gets more complicated. This is addressed in [14] by enabling midpoint nodes to optimize the path the traffic is on. Similar to [15], the authors motivate their work with the simplification of deployment of SR routes. Especially in bigger networks, the deployment is often handled in a separated control plane with a software defined networking controller, which weakens that argument a bit. Additionally, with their creative approach to reduce SR-Routes, they make the forwarding behaviour of the SR-enabled network even harder to understand. In this thesis, we will explore other advantages in minimizing the number of SR-Routes that might make these approaches more interesting and useful, while also trying to understand the SR-configurations of basic solvers better.

Micro-SIDs, which have been introduced in [16], are a great tool in reducing the size of the SRv6 header by compressing multiple SIDs into the space of a single IPv6 address. Similar to compressed SIDs (cSIDs), which have been specified in [17], they make use of the fact that a 16-byte IPv6 address is not necessary to cover the often rather small address space of a SR domain, while only requiring

a small change in the forwarding of midpoint nodes. In [18] from 2023, it is shown that more than half of the SRv6 users use these compressed SIDs, which highlights the demand for reducing the bandwidth taken up by the SRv6 header. As discussed in 3.2.2, the optimal solution can be found allowing for only one SID in most cases. Hence, even in compressed form, the impact of the SRv6 header should not be neglected.

In [2], SR-paths that visit some links between nodes multiple times are discussed. Routing loops are a danger to networks everywhere, but these special ones are shown to sometimes be necessary to reach the solution with the best MLU. This happens if they are the only possibility to use equal-cost multi-path routes, which is a routing strategy that is used by default in SRv6. With these so-called weak loops, they show how hard it is to assess the quality of a configuration just by intuition, as revisiting certain links just looks like a waste of resources at the first glance. The authors show that weak loops are abundant in SR-Paths generated by 2SR, but cannot answer the question of how many weak loops are really beneficial and what their presence reveals about a SR-configuration.

There are other options for SR-TE we did not include in our analysis, even though they are already implemented in the REPETITA framework, namely Defo-CP [19] and Column Generation for Segment Routing (CG4SR) [20], though for different reasons. Defo-CP uses constraint programming and searches heuristically for solutions. It is designed to be able to incorporate additional constraints about the network paths, but the base-case MLU optimization relevant in this thesis performed worse than SRLS in most metrics, especially runtime. It is interesting though that the heuristics are fundamentally different, as it e.g. prohibits weak loops as described in [2] completely. CG4SR on the other hand seems to be very interesting and definitely competitive, though its implementation in REPETITA did not work consistently. It claims to be as fast as 2SR without being restraint by the SID list length by using column generation, but has also no guarantee for optimality. Instead of heuristically, it can also be used to get the lower bound for the SR-TE routing problem. Notably, it is the only algorithm discussed that includes adjacency-SIDs for finding optimal solutions.

Chapter 3

Overview of Traffic Engineering with SRv6

3.1 Segment Routing with SRv6

3.1.1 IPv6 Routing

The Internet Protocol (IP) is the main routing protocol used in the Internet Layer. It is responsible for connectionless delivery of so-called packets, which is data sent from and to applications running on hosts on the internet. These hosts are connected by overlapping networks of routers, also called nodes, which communicate with each other in a decentralized manner to find the fastest way of forwarding data the correct way. This data, usually sent using a Transport Layer Protocol like TCP, is encapsulated with the IP header. The header includes the information needed for the routing process, including source and destination addresses, uniquely identifying devices across the internet. The address space is defined by the Internet protocol, but differs between the different versions: IPv4, which became a main building block of the internet after its introduction in 1974, defined an address as a 32 bit identifier, resulting in 4.3 billion possible addresses. This lead to address exhaustion rather soon causing the development of IPv6, which expanded the length of an address to 128 bits. The adaptation of IPv6 is continually increasing due to its obvious advantages, even though it started rather slow, as IPv4 is so fundamentally integrated in the internet as a whole.
Besides the expanded address space, IPv6 simplified its header significantly, as Ipv4 carried a lot of unused fields from old and obsolete parts of networking. Even though the IPv6 header is usually longer than the IPv4 header due to the length of IPv6 addresses, it can be processed easier due to reduced bloat and its fixed size of 40 bytes. Extra functionality can be implemented through additional extension headers, which are appended to the IPv6 header when their functionality is needed.

3.1.2 Segment Routing Principles

Routing with IP addresses relies on destination based routing. Only the destination address is needed for the packet to be delivered. In fact, neither the sender nor any router in between need to know anything about the network topology, but only which of their output links leads to the desired destination. Due to its simple and decentralized nature, unwanted behaviour can occur,

e.g. overutilized network links or routing loops when routers disagree on the network state. For intra-network connections, network operators might see better routing solutions than the shortest path routing, often due to good knowledge about the topology and the expected traffic, or any required network behaviour. Source routing, which will not be abbreviated to avoid confusion, needs the whole path a packet takes through the network to be defined at the source. Although this allows for complete control over the network traffic, source routing is hard to set up well and maintain, and gets only used in special cases.
Segment routing describes a solution in between the two approaches: On top of shortest path routing, individual paths can get modified by adding detours. Routers on the way can send the packet to its detour node via shortest paths, and the router used as the detour node is responsible to remove itself from the path and send the packet further towards its destination. A detour is called a segment, and thus the name segment routing. Inside a network that supports segment routing, also called a segment routing domain, segments get a specific segment identifier (SID). These segments can be chained, making the expressivity of segment routing only limited by the allowed length of the list of the SIDs.
SR is mainly used in two different ways, either on top of a MPLS network or over IPv6. Multi-protocol Label Switching uses special labels to route traffic, so routers do not have to look up IP addresses in their forwarding table but can use simple predefined routes. As IP routing can be implemented in hardware nowadays, its benefit is not necessarily worth its complex set up. SRv6 on the other hand runs over a normal IPv6 network with minimal adjustments. SIDs have to be specified initially to enable not only routers, but also links and special network functions to be used as destination addresses inside a local SR domain. These SIDs are normal IPv6 addresses, taken from an address space that is reserved for the usage with SRv6 and have only local meaning. To signify which detours are used, their SIDs get appended to the SID list inside the SRv6 header.

3.1.3 The SRv6 Header

The SRv6 header is an extension header of the IPv6 header. If a packet enters a SR Domain or is sent from within, the SRv6 header is appended to indicate the path the packet will take through the network. The destination address in the IPv6 header is replaced by the first detour the packet takes, and all concurrent destinations are carried in the SRv6 header. Like that, routers can only inspect the outermost IPv6 header to forward a packet and only routers used as a detour need to have SR capabilities. A router that is used by a detour removes its own address from the destination field in the IPv6 header and replaces it with the next entry from the SID list. In reality, this SID list has a length limit: To not exceed the Maximum transmission Unit (MTU) of the underlying Ethernet connections, the maximum size of a TCP segment has to be limited to allow for the additional space taken up by the potential SRv6 header. 3.1 gives an idea for the composition of a packet routed with SRv6. Notably, the size of the fixed part of the SRv6 header is only 8 bytes, which is small compared to the 16 byte SIDs.

3.2 Traffic Engineering

3.2.1 Maximum Link Utilization

Network Congestions lead to packet losses and generally undesirable behaviour of the network, and their prevention is a central part in wrestling the ever-growing amount of internet traffic. Shortest path routing is indifferent to the amount of traffic flowing through the network, which can lead to overutilized links between nodes, even if only a fraction of the total network capacity is used.

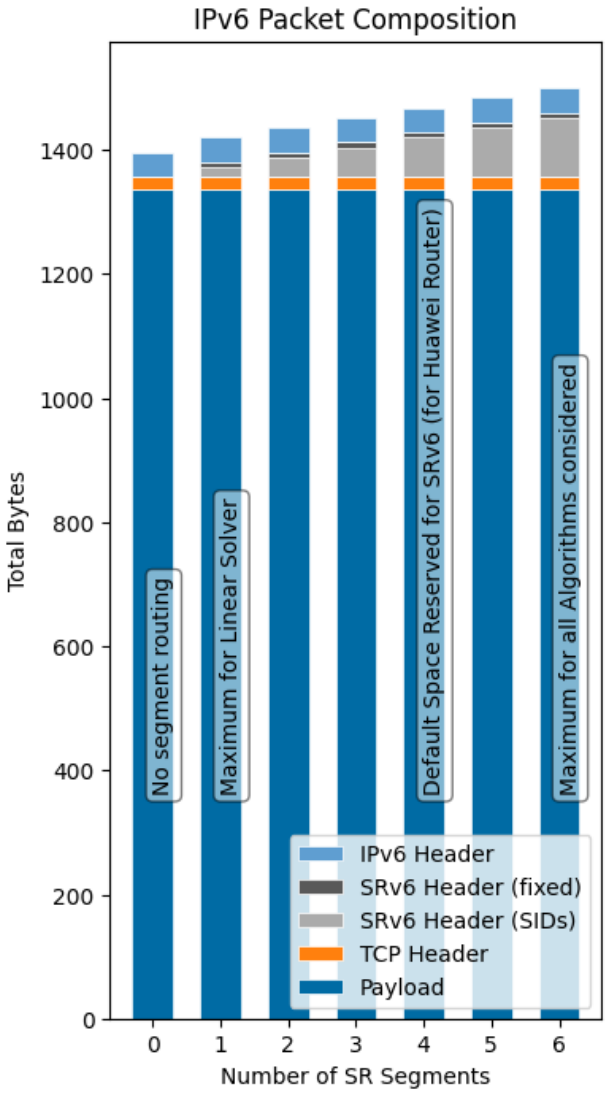


Figure 3.1

Preventing congestions is the main goal of traffic engineering with segment routing, almost exclusively done by minimizing the Maximum Link Utilization (MLU). The link utilization describes the amount of traffic flows through a link relatively to its capacity, or in other words, how close the link is to its utilization limit. By minimizing the utilization value of the maximally utilized link over the whole network, we make sure that no link gets too close to congestion. The maximally utilized link can be seen as the bottleneck of the network, as it is the most likely to congest, even though congestions could theoretically happen anywhere.

The routing algorithms compared in this thesis are all traffic aware, which means that they require a traffic matrix in addition to the topology information before they search for an optimal solution. In the general case, this is known as solving the multi-commodity flow problem. Conventional segment routing requires the flows to not be split up between different paths, which has been shown to make the problem NP-Hard [21]. In addition, SRv6 enabled routers use Equal-Cost Multi-Path (ECMP) routing wherever they can, which SR-algorithms have to account for to get to good solutions.

3.2.2 2SR: the Linear Programming Approach

[5] introduced a formulation of a traffic aware linear program to optimize the MLU by choosing up to one detour node per source-destination node pair. Consequently, the maximum SID list length is 2: one SID for the detour, one SID for the destination. Per flow, there is an integer variable for every possible detour node that is either 0 or 1, making the problem a Mixed Integer Linear Program (MILP) for which an optimal solution can be found algorithmically with an appropriate solver. In general, there is a flow for every node pair, making the size of the problem grow proportionally to the square of the number of nodes of the network.

This problem formulation has been expanded to an arbitrary number of detour nodes in [15] expanding its solution space greatly. Though it has also been shown that for most networks, this added freedom is not needed to get to a near-optimal solution. The solution quality of 2SR only starts to drop if networks get large, but at that size, the 3SR algorithm is already completely unfeasible due to memory and computation limitations. Thus, most work on better scaling algorithms focus on finding good, but not optimal solutions heuristically.

3.2.3 TLE: Adding a Second Optimization Step to 2SR

In [15], The 2SR algorithm gets adapted with an extension that minimizes the number of tunnels, i.e. the number of defined segment routing paths that differ from shortest path routing. In this thesis, we are focusing on the tunnel limit extension (TLE), that does not allow flows being split up between multiple SR paths. First, the optimization problem as formulated in 2SR gets solved. In the second optimization step, the number of detours gets minimized, with the additional constraint of the MLU value staying below the previously calculated 2SR MLU value, scaled up by multiplying with a tradeoff coefficient. If this constraint is too loose, benefits from doing traffic engineering get lost. It might even lead to the configuration with no SR paths at all, which corresponds to shortest path routing. Though, a very tight constraint might not be a problem: often, a lot of different solutions lead to the same optimal MLU, and thus even a having the original MLU without any margin as the constraint can lead to a reduction in SR paths without giving up any MLU.

The reason stated by the authors for limiting the number of SR Paths is mainly the lower cost of deploying and maintaining; Although this aspect and its relevance is not discussed this thesis, other potential benefits from this extension are presented in 4, e.g. better resilience against random changes to the network traffic.

3.2.4 SRLS: Local Search Optimization

[4] optimizes the MLU by using local search. Iteratively, it checks which small changes to the current path configuration leads to a decrease in MLU. The solutions that can be reached by a single move like inserting or removing a segment of a path are called a neighborhood. To speed up the algorithm, it is more likely to change up paths that are utilized the most. To prevent getting stuck in a local optimum, after finding no improvements in multiple neighborhoods the current solution gets saved and then perturbed. Such a perturbation is either the random replacement of a path segment or a total reset of a path, and it is needed to explore a bigger part of the solution space quickly.

This approach uses heuristics and has no guarantees of optimality. Its great strength is its speed and its ability to scale well, even to big topologies. As it works in iterations, it can be stopped at any time to get the best solution found. Through the chosen moves include the removal of segments

and paths, it happens to result in solutions with only a few segments generally.

3.3 The REPETITA Framework

The REPETITA Framework has been introduced in 2017 in [3] for repeatable testing and analysis of TE algorithms. It is a combination of already implemented segment routing algorithms, network analysis tools and a big set of network and traffic data gathered from older works. It is easy to extend with additional algorithms and data and has thus been used in numerous academic works about segment routing traffic engineering since its release.

The original 2SR algorithm as described in [5] as well as the SRLS algorithm from [4] are included, which we both used and tested extensively. While REPETITA is written mostly in Java, the algorithms can be written in any language as long as their interface is used. We implemented the Tunnel Limit Extension 3.2.3, as it was not included in the paper it was introduced in. Additionally, we adapted the existing solving algorithms to take the traffic overhead from the SRv6 header into account.

3 different public collections of topologies are included: The Rocketfuel dataset [22] that was inferred from internet measurements, a synthetic dataset used to test the DEFO algorithm [19], and a collection of real networks from the Internet Topology Zoo [23].

Chapter 4

Main

4.1 A Comprehensive Analysis of Existing Approaches

Even though TE approaches have been analyzed and compared before, the SR configurations the individual algorithms produce are still not very well understood. we ran analysis on the mixed integer linear programming optimizer 2SR, its two-step optimizer extension 2SR-TLE with two different tolerances (0.01 and 0.1), and the local search algorithm SRLS. In this section, we try to highlight different metrics that might give insights about the solution space of the problem and commonalities and differences between the approaches.

The experiments were run on all available real networks from the Internet Topology Zoo [23] available on REPETITA that have a size between 40 and 70 nodes. These sizes were chosen to highlight how the approaches fare on topologies that are rather hard, while still keeping computation time reasonable. We excluded networks that were too similar to each other, as well as networks whose Shortest Path Routing configurations were already optimal regarding MLU. 31 networks matched these requirements with 5 different traffic matrices each, resulting in 155 different problem instances. Even though SRLS' big advantage is its speed, it had the same time allocated as the baseline 2SR to make the comparison fair. As our implementation of TLE not always terminated, it could run for up to 5 times as long as the 2SR baseline. If it is terminated due to reaching the time limit, its MLU is not guaranteed to be within its given constraint.

In 4.1, we see a first comparison between 2SR and SLRS. It is clear that SRLS did not find optimal solutions for all of them, though its general distribution is comparable to 2SR. Note that all networks in REPETITA were scaled so that solving the multi-commodity flow problem leads to a MLU of around 0.9. This lower bound was reached by most of the solutions found by all different TE-approaches.

To check the main objective of the extension, we compare TLE to the other approaches in 4.2 and see how well it performs. We see clearly that even with minimal cost, most of the configured SR paths can be removed. Often, TLE matched the optimal MLU value of 2SR, while cutting down on configured SR paths significantly. Interestingly, even when its MLU was constrained to be within 0.001 of the optimal 2SR solution, TLE performed very well, reaching almost the same values as the less constrained version. We can see that SRLS also has a rather low number of configured SR paths in general; most likely, this is due to the fact that its heuristic only adds new paths with a concrete benefit, which acts as a built-in restraint. The huge range also hints at how many

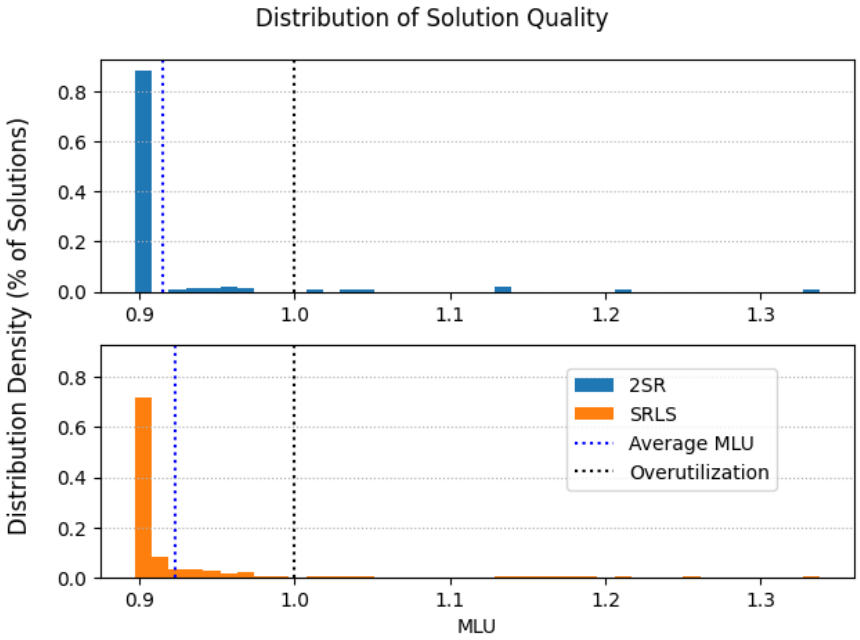


Figure 4.1

different solutions with comparable MLU there could be, and thus the potential benefit in making a reasonable choice.

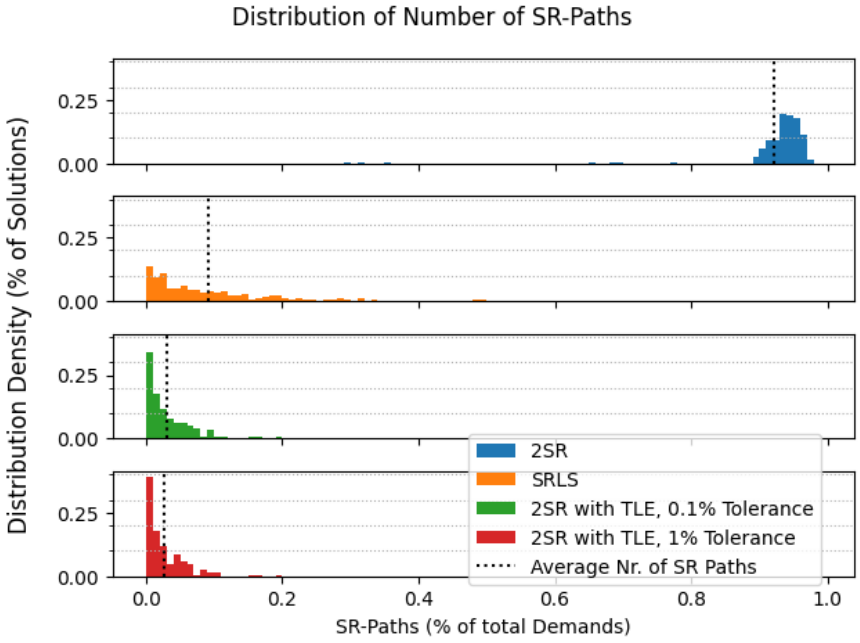


Figure 4.2

Of course, if a path is not needed to reach the optimal MLU, there is no benefit in including it. But if, how and why these unnecessary paths make a configuration worse is not directly obvious. When looking at the Average Link Utilization (ALU) in 4.3, we see the same picture: 2SR has over 30%

more ALU than SRLS, and even more than the most restrained version of TLE. We see clearly how 2SR uses a lot more resources and how this can be mitigated simply by minimizing the number of configured SR paths. Even though this huge difference in resource usage cannot be shown in the MLU, we show the benefits of being resourceful in realistic scenarios in 4.2 and 4.3.

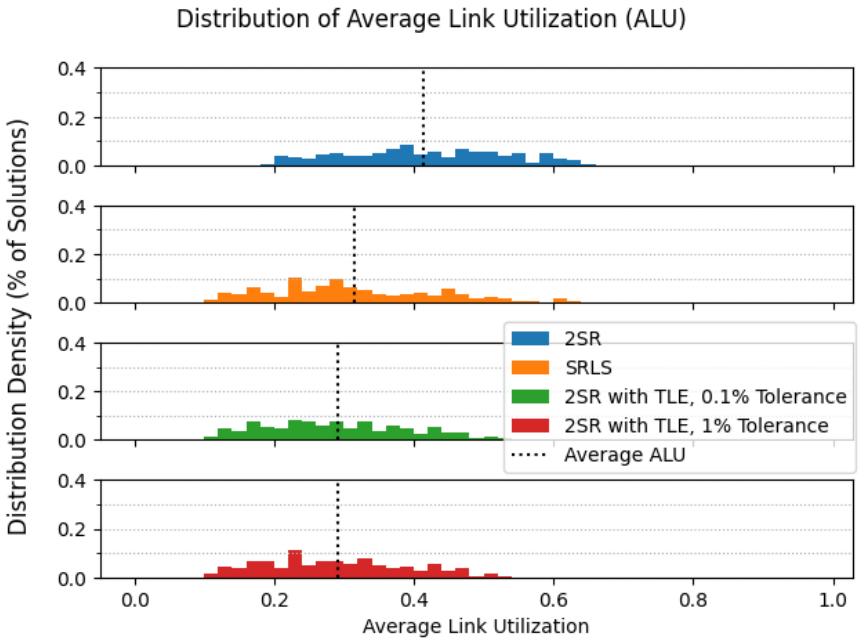


Figure 4.3

4.2 Enhancing the Existing Algorithms

Optimizing the MLU with SR algorithms is and was always done by rerouting flows of a constant size. But the size of a flow represents the amount of traffic this flow is responsible for, and by treating its size as a constant we ignore an inherent cost of segment routing: adding traffic by adding SRv6 headers to every packet that gets rerouted. To Investigate this hidden cost and possible solutions to the problem, we added this overhead to the flow simulation of REPETITA. In addition, we extended the already existing algorithms of 2SR, 2SR-TLE and SRLS to being able to optimize for this more realistic scenario.

4.2.1 Estimating the SRv6-Header Overhead

Even though network traffic is technically a quantized amount of bytes, it is sensible to treat it as a continuous value due to the amount of traffic that real networks have to handle. In the same way, we can treat the additional bytes of traffic introduced by the SRv6 header as a constant overhead factor on the traffic over a specific link, depending on the size of the header in relation to the size of a packet. Hence, we need three elements to do our calculation accurately:
(1): The SRv6 header size, depending on the number of SIDs. As discussed in 3.1.3, the SRv6 header takes up 8 bytes by itself with an additional 16 bytes per SID by design. Not clearly specified is if the empty header has to be omitted if no SR route is configured for the destination. Not specified either is if the destination address in the IPv6 header is repeated in the SID list, which would make the SID list length 2 instead of 1 if a route contains a single detour. In this estimation,

we assumed that the routers are already configured to omit the header without a configuration and to not repeat the current destination in the SID list, as someone interested in using traffic engineering would already use readily available configurations to minimize traffic. Without optimal configuration, the overhead could even be greater than in this estimation.
(2): The number of SIDs in the header. This is given by the configurations our traffic engineering algorithms produce, and has to be done per flow and per link individually. For example, if a flow has one detour, the total SRv6 header size is assumed 24 bytes on all links on the path from the source node to the detour node, and only 8 bytes on the path from the detour node to the destination node. Therefore, the traffic for that flow on the first part of the path is multiplied with a bigger overhead factor than the traffic of the second part.
(3): The size of a packet. The smaller the estimated average packet size, the bigger the proportional overhead introduced by the constant SRv6 header. With no reason to think otherwise, we assume the average packet size to be equal over the whole network.

Regarding the size of an average packet, there is no clear right answer. Clearly, it has to be somewhere between 1 and 1500 bytes, as 1500 is the Maximum Transmission Unit (MTU) of Ethernet. The Internet Mix is a standardized internet traffic profile often used for testing performance of routers or firewalls, but it has not been updated in a long time. With 340 bytes, it seems to massively underestimate the average packet size. [24] and [25] show a bimodal distribution of packet sizes, as traffic sent with TCP (Transmission Control Protocol) is comprised mostly by data packets close to the MTU of Ethernet (>1400 bytes) and acknowledgement packets (<100 bytes).
[26] from 2021 found the average layer 4 packet size to be 870 bytes in a campus network. Our analysis on data from the CESNET ISP backbone published with [27] revealed similar numbers: over all one-minute-averages, the average packet size is 843.7. The 10th and 90th percentile of one-minute-average sizes are 754 bytes and 915 bytes respectively, which shows some variance. When only including the 10th percentile with the most amount of data transmitted, the average rises slightly, again to 870 bytes. As we are interested in networks with a lot of traffic, we decide to use this number. In summary, the SRv6 overhead estimation done captures the average overhead in a typical network under high load. Though, the overhead depends on the traffic composition and surpasses our estimate in the worst case, especially if looking at short amounts of time.

4.2.2 Analyzing the Benefit of the Extension

Using the measurements done in 4.1, we compared the MLU calculated in the classical way with the MLU calculated with our more realistic approach. We see in 4.4 that configurations found with 2SR have the biggest overhead from the added SRv6 header. Note that this average measurement is heavily influenced by outliers and both TLE and SRLS have almost perfect MLU, even when looking at the 99th percentile and including the new header overhead.

How does this change when adapting our TE-Algorithms to this updated MLU? In 4.5, 2SR and SRLS are compared to their header-extended versions, and we can see that 2SR profits greatly from this change with a drop in MLU of 1.64%. SRLS improves only little; as before, the huge difference between 2SR and SRLS in average MLU is more related to the fact that SRLS has more extreme outliers.

A possible reason for the SRLS being almost unaffected by the header overhead could lie in its heuristics: While searching in a "neighborhood" for a SR path that leads to a better configuration, it is biased to reroute flows that use the most utilized link [4], and it is thus less likely that flows on the most utilized link have been rerouted themselves.

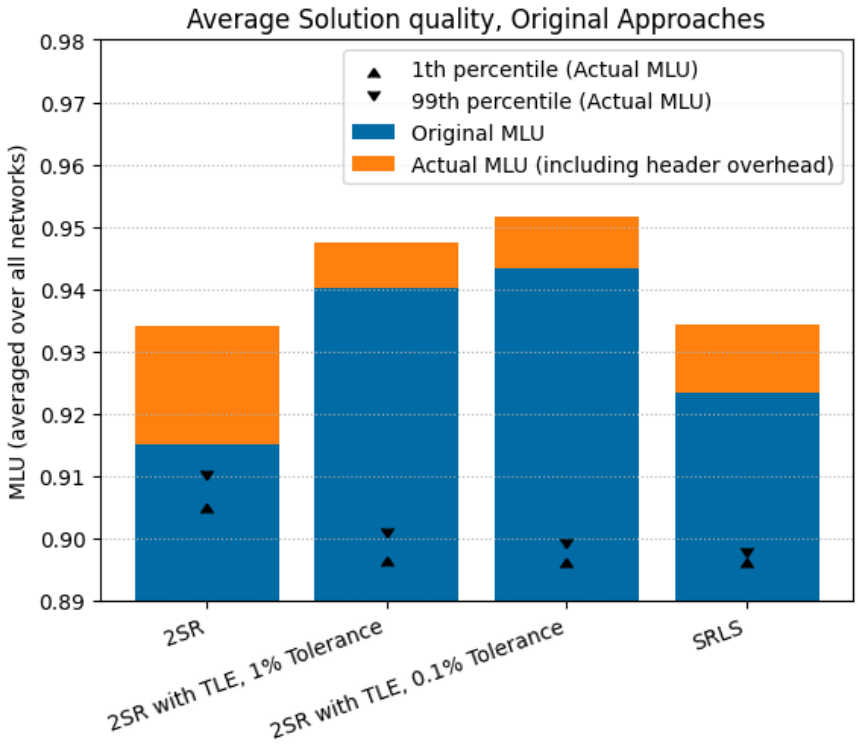


Figure 4.4

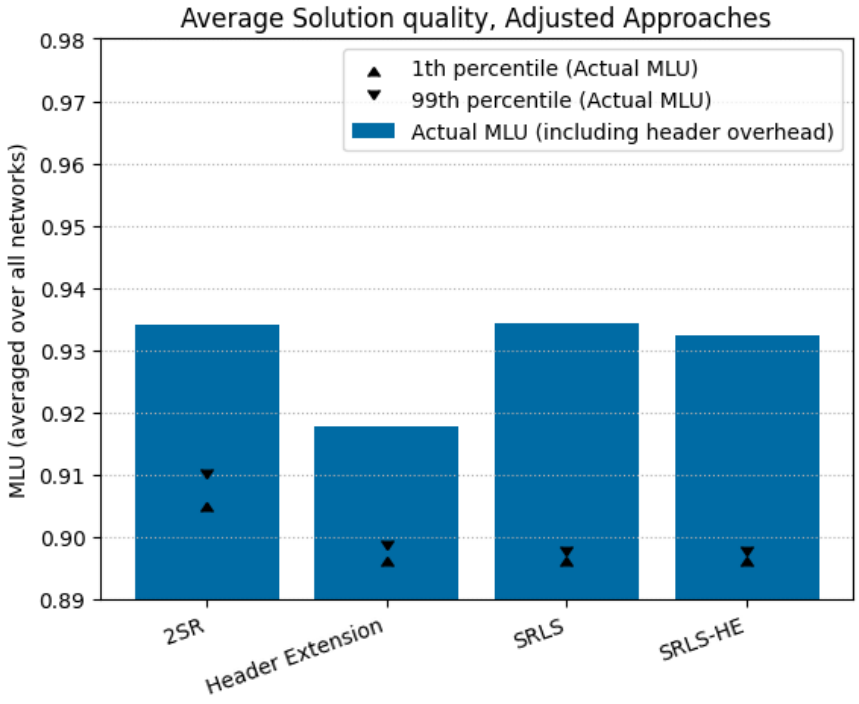


Figure 4.5

4.3 Extending the Analysis with an Emphasis on Resilience

4.3.1 Motivation: MLU is Limited

Having a minimal MLU as the optimization target is ubiquitous in traffic engineering for segment routing, and for good reasons: It is easy to understand, it is a metric that is simple enough to compute to be used as an optimization target, and lowering it clearly limits the risk of congestion when demands grow. Though, it could be worth it to question its position as the only target and investigate the relevance of other properties of configurations of SR paths.
The solution space of potential configurations of SR paths is huge, even when requiring optimal MLU. The big gains are gained by routing a few big flows that use low-capacity links, and after that, most flows can be rerouted arbitrarily without even influencing the MLU. This is seen by the big number of flows that get rerouted by the 2SR algorithm, as it has no reason to not use a detour, even if there is nothing to be gained. When penalizing SR routes using the most utilized link by using the header overhead extension as described in 4.2, the number of total rerouted flows decreases, but only by a bit. If we are ready to lower our requirements in MLU slightly, we see in 4.1 that only few of these configured paths are really necessary. This freedom highlights the potential for a lot more improvement in aspects we currently do not care for.
On another note, the MLU is not a perfect indicator of congestion risk. If we measure the quality of our SR configurations only by MLU, we limit ourselves to the limitation of this metric. It should be obvious that not only the maximally utilized link can become congested, but any other link as well, and thus it is beneficial to lower the general utilization. Also, only the relative utilization of the links is used, so a lot of information is ignored that could be relevant for predicting and thus preventing congestions when regarding network dynamics. Perhaps, the absolute utilization is also relevant.

4.3.2 Different Approaches for Estimating Resilience

To expand our view beyond MLU, a lot more metrics can be considered. If we are only interested in comparing SR path configurations after their generation, we are a lot less restrained computationally than if we had to integrate them in our solvers to begin with. We propose a few different ways to look at network resilience, explain why they could lead to different insights, and try to reason about possible improvements of our current TE algorithms.

The MLU value can be seen as the limit the total network traffic can grow until the first link gets overutilized. This approach got taken one step further in [6] by introducing the growth vector, which describes for every flow how much it can grow if the whole network growths with it.
This is done in iterations: first, all flows grow until a link gets to full utilization, then, all flows that now cannot grow any more get blocked, their growth value gets fixed, and all unblocked flows grow again until the next link gets to full utilization. Consequently, the smallest entry in the growth vector is the inverse of the MLU; if the maximally utilized link has a relative utilization of 0.5, all flows that use that link can grow by a factor of 2 at the same time. The biggest value of the growth vector correlates to the last flow that gets blocked after all other flows have stopped growing. Typically, the smallest flows in the network get blocked last, and thus the biggest entries of the growth vector can get arbitrarily big.
There is a lot of information packed into this resulting growth vector. An obvious benefit of this method is the incorporation of flow information. Every metric that focuses only on the utilization of links cannot capture if the unused capacity of the network can actually get used with the current segment routing configuration. Though, from the definition of the growth vector, there is no

apparent goal, only that big numbers are better and a lot of additional interpretation is needed. We know that we cannot use the minimum value, as that is just the MLU and we have not gained any additional information. Also, we cannot use the maximum value, as that value can get arbitrarily big if the flows can get very small, which also makes the average of the growth vector rather meaningless. We chose to focus on the median value, as a representation of the general growth potential without being too much influenced by outliers. Other approaches could be considered to convert this growth vector into a single, comparable value: weighting values by either their closeness to congestion or by the size of the corresponding flows seem like sensible approaches.

The MLU can also be seen as a direct measure of how well the network can handle traffic that deviates from the given traffic matrix, be it through normal traffic fluctuation or due to a traffic matrix that does not represent actual traffic. If SR configurations are optimized only by looking at the MLU, they can only be as good as their underlying traffic matrix. These variations will always be present, but we can try to minimize their impact.
As a first step, it is natural to minimize wasted resources, so that there is enough free capacity in the network to tolerate additional unforeseen traffic. But still, it really matters where those resources get wasted: the first 20% of utilization on a given link is way less relevant than the last 5%. Even though average link utilization is a relevant intuitive metric, it does not include this prioritization of links that are close to congestion. Thus, we need a way of non-linearly weighting free capacity depending on if it is needed in case of unforeseen traffic variance.

One simple way we chose and implemented in our analysis is a probability based resilience measurement. It models flows as normal distributions with mean equal to their size given by the traffic matrix. Additionally, it assumes the variance to be linearly dependent on the size of the flow, leading to flows with the general distribution of $\mathcal{N}(\mu = m, \sigma^2 = var * m)$. This simplifies calculations significantly, as the sum of two smaller flows has the same shape as one bigger flow. One could easily imagine more sophisticated probabilistic models; we chose a normal distribution because it is computationally simple and easy to understand, as well as being always a good first choice when modeling unknown uncertainty in the real world.
On the link level, it is enough to look at the traffic over that link as one single flow; as the flows are assumed to be independent, the sum of all normally distributed flows has itself a normal distribution with the sums of the means as its mean and the sum of the variances as its variances, which is assumed to be linearly dependent on its mean.

Now, we have a probability of congestion for every link that only depends on its relative utilization and on our chosen variability factor. This variability factor can be used to influence how small flows can be to still be relevant. If this variability factor is very high, almost all links have a similar chance of congestion independent of their utilization; if the variability factor tends to 0, only the current utilization matters with no regards to randomness. For our analysis, we chose the variability factor to be 0.5.
Although flows can be assumed to be independent of each other, these congestion probabilities per link are not independent as the same flow usually traverses multiple links. So, we still have the problem of uniting all congestion probabilities to one network-wide value that can be evaluated and compared. One possibility is to calculate Boole's inequality, also known as the union bound, which says that the probability of at least one event happening cannot be greater than the sum of the probability of all events. This union bound of the probability of congestion can be evaluated and compared, but cannot be treated as a probability anymore, made evident by the fact that its value often exceeds 1.
Another way to calculate a network-wide value is to take the sum of the probabilities of links

not being congested, as that represents the probability of the whole network not being congested. Of course, this probability is wrong as we cannot assume independent probabilities of congestion, but if the inaccuracies are small enough, this metric could still add insight into the resilience of a network.

To evaluate these rather simple, but computationally reasonable metrics, we also implemented a sampling-based resilience measurement. For those, we generated 1000 variations of a given traffic matrix, with each flow of size m being represented as a normally distributed random value with $\mathcal{N}(\mu = m, \sigma^2 = var * m)$, with $var = 0.5$ as in the analytical resilience measurement. For a given SR configuration found given an initial traffic matrix, this state of the network is evaluated for all variations of the traffic matrix without changing the configured SR paths. Especially interesting as a measurement of resilience is the average increase (or decrease) in MLU, as an MLU over 1 signifies a case of congestion.

4.3.3 Relating Sampling Based Resilience back to Simple Metrics

To evaluate the soundness of our estimation metrics, we run our analysis on 10 networks spread evenly between 25 and 75 nodes. For every network, we ran the basic 2SR, the TLE with two different tolerances, and 10 instances of SRLS, all without the header extension, leading to 13 SR configurations per network. For one of the ten networks (SURFNET), SRLS did not manage to get to a competitive MLU; we decided to leave out that network in our resilience analysis, as that outlier tells us more about the general capabilities of SRLS than the resilience of its solution.
For the sampling-based congestion prediction approach, we generated 1000 traffic matrix variations per network, and for every SR configuration we calculated the change in MLU, averaged over all 1000 variations, and the percentage of congested network variations, i.e. variations with MLU over 1.

When averaged over all variation, the MLU rises for all SR configurations. This is expected: the MLU gets optimized for a single traffic matrix, and if the traffic starts to vary, the SR configuration does not fit the problem anymore. Note that this happens even though the average size of the flows stays the same. Thus, we can expect that the MLU of the shortest path's configuration (i.e. no configured SR paths) would not change after introducing variance, but just because it started in a completely unoptimized place.

Clearly, not all solvers are influenced the same by the traffic variations as seen in 4.6: Even though the 2SR configurations start with the best MLU, they drop to the last place after introducing variance. SRLS outperforms it by 0.85%, and the most restrained version of TLE even by 1.77% MLU. One rather obvious reasoning for this could be that SRLS introduces too many unnecessary detours and thus increases the general load on the network, which becomes costly as soon as the traffic matrix starts to change. Both versions of the 2SR with tunnel limit extensions react better to the changes than the SRLS, which signifies that the cause of the added resilience could lie in the number of configured SR paths.

This clearly shows that there could be benefits in minimizing the number of configured SR paths. As SRLS is algorithm that uses heuristics and produces a different configuration every time it runs, it would be great to have a metric that helps pick the most resilient configuration of different SRLS runs with comparable MLU. To analyze the most simple version of that problem, we run 100 instances of SRLS on one of the networks used before (HiberniaGlobal) with different timing constraints. If the number of configured SR paths is the main factor influencing the resilience, we should see a direct correlation as both solver and network are fixed.

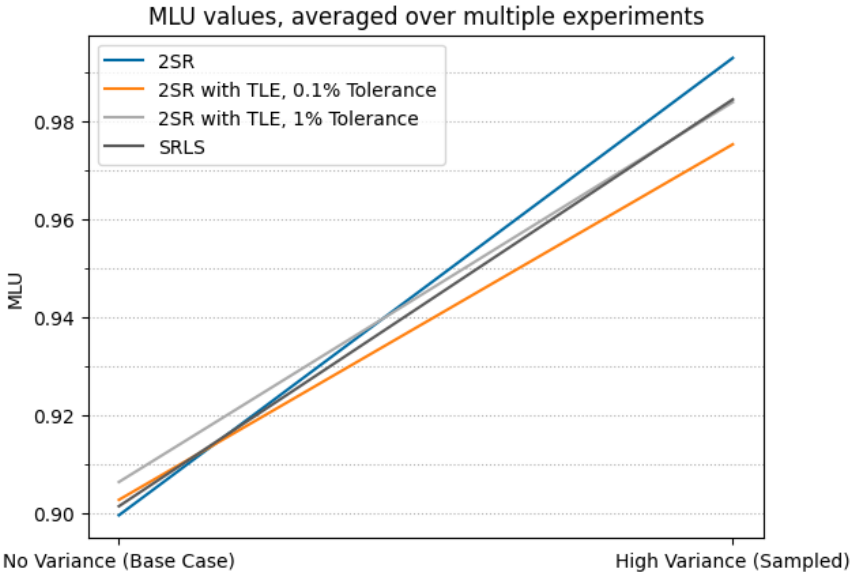


Figure 4.6

In 4.7, we see that that is not the case. Even though optimizing for the number of SR Paths had clearly a positive impact on this measure of resilience, this cannot be used for comparing intra-network differences between configurations. The same is true for the average LU and our simplified analytical congestion prediction, which can be seen in 1 and 2 in the appendix.

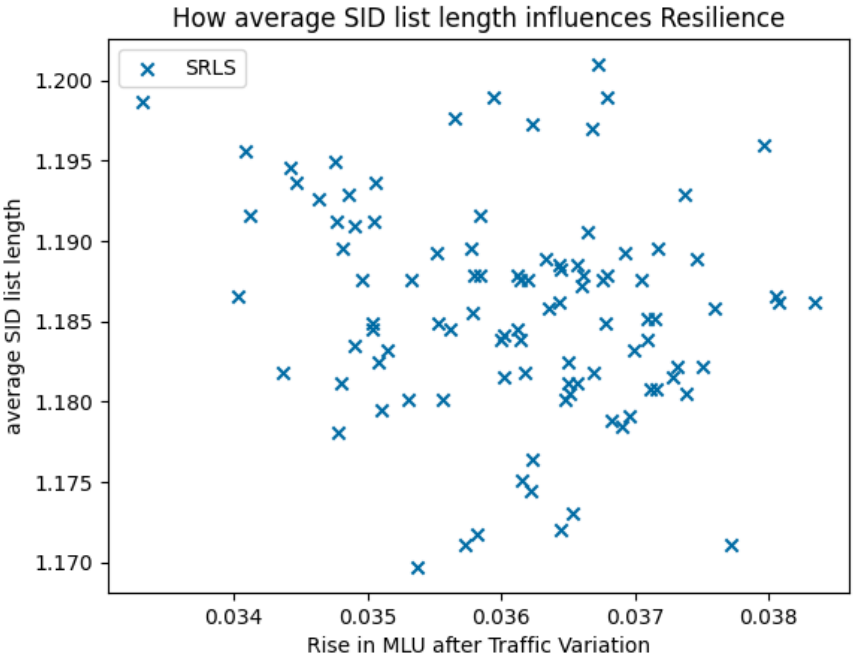


Figure 4.7

Chapter 5

Outlook

The analysis done in this thesis can still be improved and expanded upon. On one hand, more experiments could be done to see if the findings are related to the size, interconnectivity and other aspects of the network as an additional way to find advantages and disadvantages of different approaches. A better grasp as to which networks are hard to optimize could in turn lead to insights in good network configuration under SR. Also, new approaches like CG4SR can be tested with the same framework.
On the other hand, the analytical congestion prediction metric would improve a lot by removing the inaccurate assumptions relating to conditional dependence, and could result in a metric that is meaningful, concise, interpretable and deterministic. More intricate mathematical methods would be needed, but if its computational cost stays small, it could be used as an optimization target instead of the MLU.

A simple, but promising approach for solving any upcoming problem is to expand two-step optimization to other optimization targets. While it is easy to adapt the existing algorithms, the work lies mostly in justifying the loss of MLU with other benefits, as we tried to do in this thesis with the added resilience after optimizing for the number of configured SR paths. Rather obvious choices to try are the Average Link Utilization or the Nth most utilized link, as an extension to the MLU similar to the growth vector.

Rather than focus on technical advancements for getting slight improvements in MLU, it could be far more helpful to clearly model the problem SR should solve.
Should it lower latencies by preventing delays from congestions in the network? Then, it is clearly not worth it to lower the MLU from 0.51 to 0.50 by sending traffic over a path that takes twice as long.
Should it reliably lower the probability of congestion? Then, it has to be analyzed where and why congestions happen in reality, which parts of the traffic matrix can be trusted, and how to model this uncertainty in the TE algorithms.
Should the all flows be able to grow evenly without a lot of uncertainty? Then, optimizing e.g. the growth vector might be a better fit than the MLU.

A possible lack in theory is the interplay between SR-TE algorithms and other network mechanics. Only little work has been done investigating the consequence of fast rerouting with TI-LFA regarding TE objectives. TCP congestion control has not yet been connected to SR-TE, and it would be interesting to see if and how the presence of self-regulating flows could change optimal routing

configurations. Also, using multipath TCP in combination with SR-TE like in [28] could lead to better performance in networks that are hard to optimize.

Segment routing is a very powerful tool that gets used successfully in the real world, and is adaptable enough to solve a broad range of problems. At the moment, we believe it to be limited rather by the understanding of the problems to be solved than by its capabilities.

Bibliography

[1] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, and D. Voyer, "Ipv6 segment routing header (srh)." RFC 8754, Mar. 2020.

[2] A. Brundiers, T. Schüller, and N. Aschenbruck, "On the benefits of loops for segment routing traffic engineering," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pp. 32–40, 2021.

[3] S. Gay, P. Schaus, and S. Vissicchio, "Repetita: Repeatable experiments for performance evaluation of traffic-engineering algorithms," 2017.

[4] S. Gay, R. Hartert, and S. Vissicchio, "Expect the unexpected: Sub-second optimization for segment routing," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, 2017.

[5] R. Bhatia, F. Hao, M. Kodialam, and T. Lakshman, "Optimized network traffic engineering using segment routing," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 657–665, 2015.

[6] A. Hassidim, D. Raz, M. Segalov, and A. Shaqed, "Network utilization: The flow view," in *2013 Proceedings IEEE INFOCOM*, pp. 1429–1437, 2013.

[7] A. Bashandy, S. Litkowski, C. Filsfils, P. Francois, B. Decraene, and D. Voyer, "Topology independent fast reroute using segment routing." RFC 9855, Oct. 2025.

[8] L. Roelens, Ó. G. d. Dios, I. d. Miguel, E. Echeverry, and R. J. D. Barroso, "Performance evaluation of ti-lfa in traffic-engineered segment routing-based networks," in *2023 19th International Conference on the Design of Reliable Communication Networks (DRCN)*, pp. 1–8, 2023.

[9] T. Schüller, N. Aschenbruck, M. Chimani, and M. Horneffer, "Failure resiliency with only a few tunnels – enabling segment routing for traffic engineering," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 262–274, 2021.

[10] S. Martin, Y. Magnouche, P. Medagliani, and J. Leguay, "Alternative paths computation for congestion mitigation in segment-routing networks," in *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 1159–1164, 2024.

[11] B. Ren, D. Guo, Y. Yuan, G. Tang, W. Wang, and X. Fu, "Optimal deployment of srv6 to enable network interconnection service," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, pp. 120–133, 2022.

[12] C.-H. Lung and H. ElBakoury, "Exploiting segment routing and sdn features for green traffic engineering," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, pp. 49–54, 2022.

[13] D. Otten, M. Ilsen, M. Chimani, and N. Aschenbruck, "Green traffic engineering by line card minimization," in *2023 IEEE 48th Conference on Local Computer Networks (LCN)*, p. 1–8, IEEE, Oct. 2023.

[14] A. Brundiers, T. Schüller, and N. Aschenbruck, "Midpoint optimization for segment routing," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, pp. 1579–1588, 2022.

[15] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, "Traffic engineering using segment routing and considering requirements of a carrier ip network," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1851–1864, 2018.

[16] A. Tulumello, A. Mayer, M. Bonola, P. Lungaroni, C. Scarpitta, S. Salsano, A. Abdelsalam, P. Camarillo, D. Dukes, F. Clad, and C. Filsfils, "Micro sids: A solution for efficient representation of segment ids in srv6 networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 774–786, 2023.

[17] W. Cheng, C. Filsfils, Z. Li, B. Decraene, and F. Clad, "Compressed srv6 segment list encoding." RFC 9800, June 2025.

[18] R. Mota, "Acg segment routing survey," 2023.

[19] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 15–28, 08 2015.

[20] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, "Cg4sr: Near optimal traffic engineering for segment routing with column generation," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1333–1341, 2019.

[21] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 691–703, 1976.

[22] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 2–16, 2004.

[23] 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, pp. 1765–1775, 2011.

[24] D. Murray and T. Koziniec, "The state of enterprise network traffic in 2012," in *2012 18th Asia-Pacific Conference on Communications (APCC)*, pp. 179–184, 2012.

[25] D. Murray, T. Koziniec, S. Zander, M. Dixon, and P. Koutsakis, "An analysis of changing enterprise network traffic characteristics," in *2017 23rd Asia-Pacific Conference on Communications (APCC)*, pp. 1–6, 2017.

[26] P. Jurkiewicz, G. Rzym, and P. Boryło, "Flow length and size distributions in campus internet traffic," *Computer Communications*, vol. 167, pp. 15–30, 2021.

[27] T. Benes, J. Pesek, and T. Cejka, "Look at my network: An insight into the isp backbone traffic," in *2023 19th International Conference on Network and Service Management (CNSM)*, pp. 1–7, 2023.

[28] A. A. Barakabitze, L. Sun, I.-H. Mkwawa, and E. Ifeachor, "A novel qoe-centric sdn-based multipath routing approach for multimedia services over 5g networks," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2018.

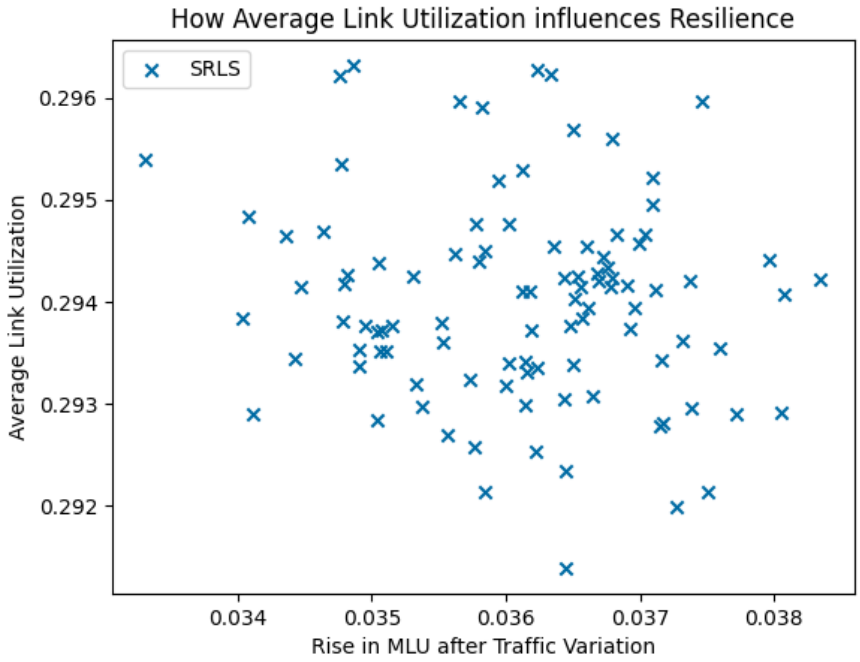


Figure 1

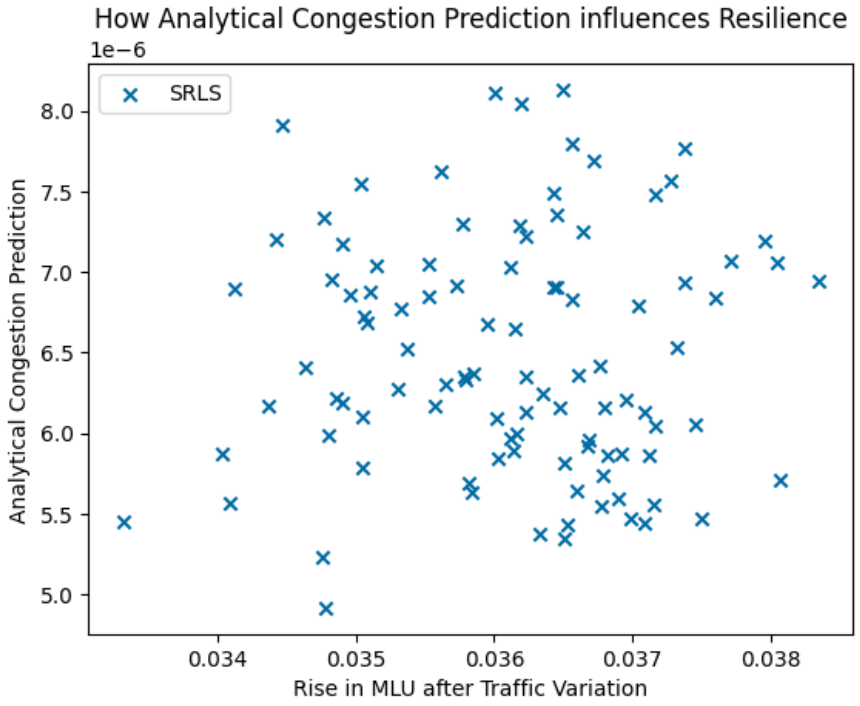


Figure 2