Bachelor Thesis Green Routing

Department of Computer Science

OST – University of Applied Sciences

Campus Rapperswil-Jona

Spring Term 2022

Authors:	Jonas Hauser, Pascal Schlumpf
Advisor:	Prof. Laurent Metzger
Co-Advisor:	Julian Klaiber
Project Partner:	Cisco Systems represented by Francois Clad
External Co-Examiner:	Laurent Billas
Internal Co-Examiner:	Prof. Stefan Keller

Abstract

Traditional routing protocols and techniques are often used in today's networks, and their basics were generally established before the millennium. In recent years, the network area has not experienced the same level of fast transformation as other IT industries. With the development of the digital world and the introduction of new industries and technologies like 5G and cloud computing, the volume of data transferred through networks today is massive and will continue to expand in the future. Modern networks must not only deal with an unprecedented amount of data transmissions, but many new requirements have emerged in order to meet client demands. In our time with climate change a new requirement on the energy efficiency of routing has emerged.

The latest estimates for the ICT sector indicate emissions of around 1.4Gt of CO2 per year. Internet backbone networks are responsible for six percent of this ecological footprint. The growing bandwidth creates new opportunities to consider other metrics and aspects in addition to the traditional ones which mostly only tend to use more and more bandwidths.

This thesis is to look for a solution to implement a green routing approach, where in a network the most ecological paths are to be computed. The solution should be able to compute paths efficiently underlying a defined green index based on sensor data from routers as well as external factors like the source of electricity or the cooling used in the datacenter. It should be possible to view the network over a simple web interface and compare different scenarios. Additionally, it should be possible to deploy the calculated green route on the network.

The application can be accessed over a frontend where the synchronized network is displayed, and the calculation of a green route can be executed. It is possible to select the metrics used for the green index beforehand. The underlying calculation of the best paths based on the green index has been implemented by a Green SR-App software in form of a REST-API. This backend API, which is written in modern GoLang, can synchronize all network data via the Jalapeño API Gateway. It can react on topology changes, process and store the received data for future statistical analysis and then calculate the best paths over a predefined period of sensor data based on Yen’s k-shortest paths algorithm. If desired, the generated green route can then be deployed on the network. The software is designed to be very performant despite very large networks of up to 1000 routers with many times more links in between. It is also possible to calculate the fastest route over the network, which can then be used to compare the greenest path to the fastest path and their metrics.

Management Summary

Initial Situation

With the growing urgency to act against climate change and simultaneously growing demand of bandwidth and throughput a new mindset regarding routing protocols needs to develop. It should be possible to route packets with a different metric than just IGP link costs. With the new segment routing protocol in combination with a SR-App it is possible to route packets with any algorithm and therefore allows to explore new use cases. To get back to climate change, it is now possible to take any metric as a basis for a route calculation. With this new possibility it would be useful if you could route packets along the path with the least environmental impact. Especially in situations where high-speed networking is not needed certain customers would choose a greener path over the fastest path. This is the idea behind this thesis which is a follow-up thesis from the Green Routing thesis written in the autumn term of 2021.

This thesis is to extend the prototype of a green routing approach, in which the most ecological paths in a network are computed. The solution should be able to compute paths efficiently by using a defined green index based on sensor data from routers as well as other metrics such as the efficiency of the datacenter or the source of the electricity.

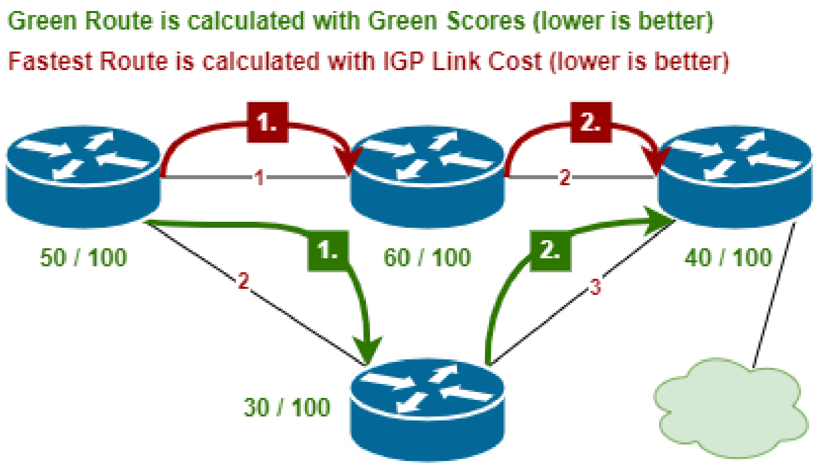

Green Route is calculated with Green Scores (lower is better)
Fastest Route is calculated with IGP Link Cost (lower is better)
1.
2.
1
2
50 / 100
60 / 100
40 / 100
30 / 100
1.
2.
2
3

Figure A.1: Network with Green Route compared to Fastest Route [1]

Procedures and Technology

For the project inception we met with our supervisor and his team to discuss the next steps as well as the expectations they have for the product at the end of our bachelor thesis. Based on this meeting we expanded the use cases and created the product backlog.

In the elaboration phase we researched the best way to implement multiple equally viable paths, what kind of metrics we could use and how to create a green index based on these metrics. One option was to switch all graph data to a separate ArangoDB which allows us to calculate k-shortest paths on the data. However, this idea was rejected, and we decided to use an existing library which offered Yen's k-shortest path algorithm. This approach was leaner and posed a smaller risk than the switch of the whole database. Some proof of concepts were also undertaken during the elaboration phase. Namely for the frontend, for the configuration of the routers as well as for the usage of the subscription service.

In the construction phase the project was continued on the foundation of the project thesis. The continuation went flawlessly, and progress was achieved swiftly. The main new functionality developed were the frontend, the calculation of multiple green routes with multiple metrics, and the configuration of the network.

Results

We were able to fulfill all the mandatory use cases, which includes the ability to calculate multiple green routes based on the synchronized data from the Jalapeño API Gateway. If desired, it is possible to use the subscription service from the Jalapeño API Gateway to react on topology (? Font) changes. A user is able to individually choose the metrics used for the next calculation. To prove that the green route takes a different route, we developed not only the green route calculation but also the fastest route calculation. On average, for a network with 1000 nodes and 25000 links, the calculation is still completed in less than 2.5 seconds. It is then possible to deploy the SR-Policy created from the green route to the necessary routers over SSH. We built a frontend with a view in which the selected green route and the selected fastest route can be displayed.

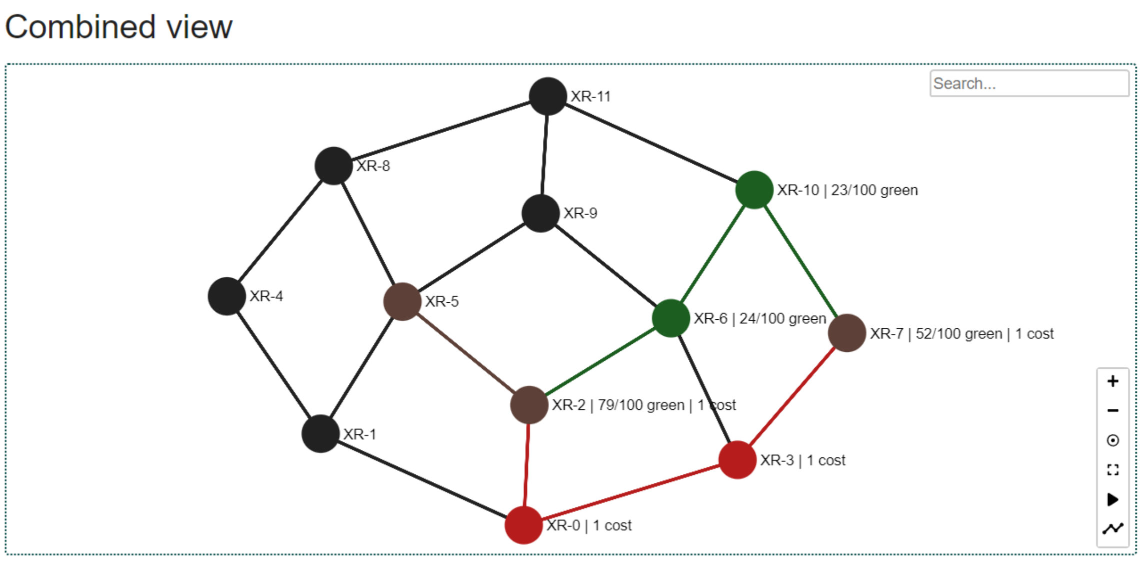


Figure A.2: View of topology with highlighted paths in the frontend [1]

We extended our mocks of the data access layers per domain to be able to write integration tests without having an in-memory or local database running. We achieved a test coverage of over 80%. An additional aspect is the deployment to the Kubernetes cluster of the INS which allow for an easy setup of our application.

Outlook

The first steps to improve the application would be to fix the various shortcomings of the Jalapeño API Gateway and some bugs of Jalapeño itself. This would make the code a lot leaner and more easily maintainable. One thing which was not possible until now, is to test the application with a real network and observe how it behaves. Regarding the configuration it would be preferable if the process could be switched over to gRPC with Yang Models instead of multiple SSH commands. Not only would this be faster but also easier to maintain with a clearly defined interface.

Acknowledgments

We'd like to express our appreciation to the people listed below for their assistance and/or supervision during our bachelor thesis.

Laurent Metzger is the person who came up with the idea for this project and gave us the chance to continue our term project work. We thank him for his constant helpfulness and the always very pleasant meetings.

Julian Klaiber who was co-supervisor of Laurent Metzger and supported us with valuable feedback and ideas. He was also always open to support us in areas where our knowledge was limited.

Michel Bongard who was co-supervisor until the project week four and supported us with valuable feedback and ideas during the initial project time.

Severin Dellsperger who is a member at the INS and has big knowledge in Segment Routing. He supported us with feedback and was always available for questions and help.

Matteias Collet who was proofreader of our documentation and also gave useful feedback on the content.

Contents

A. Technical Report

Change history

Version	Date	Changes	Responsible
1.0	24.06.2022	Finished the final technical report.	Jonas H. and Pascal S.

Contents

1. Introduction

This document is intended for engineers in the field of computer science. A basic understanding of networking and software engineering is required.

The progress of this thesis was built on the results of the project term of the autumn term 2021. Continuing on the previous achievements enabled us to expand and improve the already existing code and use cases. [1]

1.1 Thesis composition

This thesis follows the guidelines set by the Eastern University of Applied Sciences and is organized into three main parts.

1.1.1 Technical report

There are three chapters in the technical report.

The first chapter (part A) gives a brief summary and introduction to the work. If any techniques were employed, they are described. The goals and tasks are determined by the problem definition and solution, as well as the methods employed. The outcome of this project is described in the results and discussion chapter, and the implementation is reflected in rudimentary form. Important sections for the conclusion can be found in the third and final chapter. For each section of this thesis, the experiences and lessons learned are examined. In addition, possible next steps are outlined, as well as potential enhancements and an outlook into the future.

1.1.2 Project documentation

The second chapter of the report (part B) covers the entire study, including how the results were achieved.

The requirements are specified in the first chapter as use cases and divided into functional and non-functional objectives. The second chapter covers all aspects regarding project management. The methods employed, as well as their important data, are detailed, including milestones, schedules, meetings, responsibilities, risk management, and other elements. It is discussed how we intend to build this thesis after the project management chapter. The principles, version control, quality metrics, error handling, development environment, and continuous integration and delivery (CI/CD) are all described. These measures should help the team during the development. The architecture and design specifications, as well as a domain analysis, are described in the following chapter.
The declaration of independence, the terms of use of this work, and a reference to the meeting minutes are all located in the last three chapters.

1.1.3 Appendix

In the appendix (part C), supporting documents and figures can be found.

1.2 Existing Research

In the last few years, a lot of research has been done in the area of segment routing, especially in regard to increasing the efficiency of datacenters.

The research can broadly be grouped into the following three topics:

1.2.1 Standby mode

Multiple studies performed some research with the goal to increase the efficiency of a datacenter by turning off parts of a network to save power. This way there is only ever so much capacity as needed and a lot of power can be saved. This is achieved with segment routing by only utilizing a subset of nodes and edges to be able to put the remaining ones into a sleep mode [2–5]

1.2.2 Load spreading

Other studies approached the topic from a different angle by preventing overloading certain links and therefore wasting energy. This is also achieved with segment routing, although they still use label-based routing. With their method, traffic is spread over a bigger area to achieve a better overall capacity utilization, leading to a lower overall power consumption. [3, 4]

1.2.3 Green routing

In our research we only found one other study going into a similar direction as our thesis. In that study the author also defined what a green router is and how to route traffic along the greenest path. The routing is still done with **MPLS** and not with the newer **IPv6** segment routing possibilities. [6]

1.3 Techniques

In this chapter we describe different technologies and systems we encountered or used during our thesis. We ~~will~~ explain from a high-level perspective how these different techniques work and how they helped us achieve our goal.

1.3.1 Segment Routing over IPv6

The term green routing is used in various sectors to indicate a greener alternative to the current standard. In networking this has been lacking so far. The reason being, that the existing routing protocols couldn't provide the needed flexibility to allow for such routing algorithms. This has been changed with the introduction of segment routing (SR). Specifically, SR over the *Multi Protocol Label Switching (MPLS)* plane and SR over the *Internet Protocol Version 6 (IPv6)* plane, also called SRv6. In this section we will focus only on the latter, namely SRv6. We explain how this protocol is designed and how it ~~all~~ works. The explanations are based on the following RFCs from the IETF:

1. RFC 8402: Segment Routing Architecture [7]
2. RFC 8986: Segment Routing over IPv6 (SRv6) Network Programming [8]
3. IETF Segment Routing Policy Architecture draft-ietf-spring-segment-routing-policy-22 [9]

Concept

SRv6 SID Format

SRv6 segments are identified using segment identifiers (SIDs) encoded as IPv6 addresses. An SRv6 SID consists of three parts expressed in the *Locator*:*Function:Args* format as seen in Figure 1.1

Locator identifies a node and is used to route packets through the network. It needs to be distributed in the network to guarantee connectivity

Function identifies an instruction bound to the node that generates the SRv6 SID

Arguments occupies the least significant bits of the IPv6 address and can be used to define packet flow and service information

The maximum length of a SID is, equal to a standard IPv6 address, 128 bits. It is not necessary to use all the bits and the arguments are also optional.

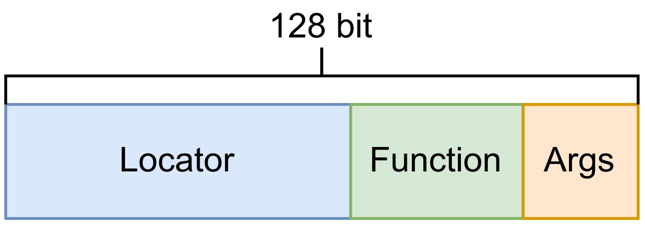


Figure 1.1: SRv6 SID [10]

Segment Routing Header

IPv6 packets are made up of an IPv6 header, 0 to N (N ≥ 1) extension headers, and a payload. The Segment Routing header (SRH) [11] is an extension header that is added to IPv6 packets in order to implement Segment Routing IPv6 (SRv6) using the IPv6 forwarding plane. It specifies an IPv6 explicit path and stores IPv6 segment lists that function similarly to SR-MPLS segment lists. The ingress appends an SRH to each IPv6 packet, allowing transit nodes to forward the packets based on the SRH's path information. You can see the details of the head in Figure 1.2 [8, 11].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Hdr Ext Len  | Routing Type  | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Last Entry   |     Flags     |              Tag              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[0] (128-bit IPv6 address)             |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
                              ...
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[n] (128-bit IPv6 address)             |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                                                             //
//         Optional Type Length Value objects (variable)       //
//                                                             //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1.2: Segment Routing header [11]

The processing of a packet, which is routed through a Segment Routing Domain, can be cut down as follows: The header contains the various Segments that must be completed during the path traversal in the Segment List. Hereby the last Segment List entry corresponds to the Segment that should be executed first. In addition, the header contains the Segments Left field, which indicates how many instructions remain to be executed. This field refers to the last Segment List entry at the start. This value is reduced by one as soon as the first Segment is done. Therefore, the field refers to the next Segment to be processed. During the next step, the Active Segment is copied into the IPv6 header's destination address field. The packet is then sent on its way to the next appropriate node based on the destination address information. If the packet reaches an intermediate node, which is not part of the segment routing domain or does not match the destination address, it is redirected based on its routing entries. The instruction can be defined as completed as soon as the packet reaches the node responsible for the Active Segment. This process will now be repeated for each entry in the segment list until it reaches the egress node, where the packet leaves the segment routing domain. This way it is possible to define any path through a network based on a self-defined algorithm, which is then documented and executed through the Segment Routing Header.

Example

To get a better idea how the whole process works, it is illustrated in Figure 1.3. The currently active segment id is marked in orange. In the top right corner, the field Segments Left can be found, starting with three. The whole series of events is started the moment a packet enters the segment routing domain via the ingress node. Said node appends the segment routing header with the segment id list to the packet and sends it to the first destination according to the list, which is R3. After the arrival of the packet at R3 the index is moved down by one and the new SID is written into the destination address field. After passing R6 it takes the shortest path to R10 which leads via R7 and R8, which themselves just route the packet according to the destination address field. At the final destination, the segment routing header is removed and the packet leaves the segment routing domain via the egress node.

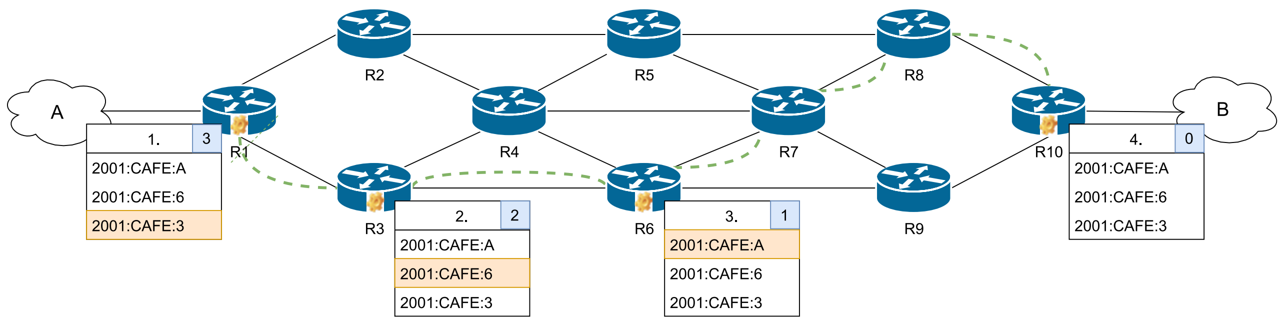


Figure 1.3: Example Segment Routing SRv6 [10]

SR-Policies

An SRv6 Policy is a collection of paths comprised of a segment list. Each SID list defines a point-to-point path from the source to the destination, instructing a device to use the defined path rather than the shortest path computed by an IGP to forward traffic. The header of a packet directed to an SRv6 Policy is extended with an ordered list of segments associated with that SRv6 Policy, allowing other network devices to execute the instructions encapsulated in the list. In other words, a SR-Policy is a configuration that ensures that packets with the same parameters are assigned to the correct Segments.

A SR Policy is identified by three elements [9]:

- **headend** is the IPv4 or IPv6 address of the node where the policy is instantiated (the ingress router)
- **color** is a numeric value to differentiate between different policies with the same endpoint
- **endpoint** is the IPv4 or IPv6 address of the destination of the policy (the egress router)

By utilizing this technique, individual packets can be treated differently based on their characteristics. As a result of the introduced tuple (headend, color, endpoint), each policy is distinct. Figure 1.4 demonstrates two distinct policies with separate meanings. Both policies are implemented on the R1 headend. The red policy should direct packets to the destination node R10 at the lowest possible cost. The tuple (R1, 10, R10) uniquely identifies this policy, with the 10 representing the numeric value for the policy of minimal cost. The green policy is distinguished by its distinct tuple (R1, 20, R10). Its purpose is to direct traffic to the destination router R10 via the green route, i.e. the path with the lowest environmental impact.

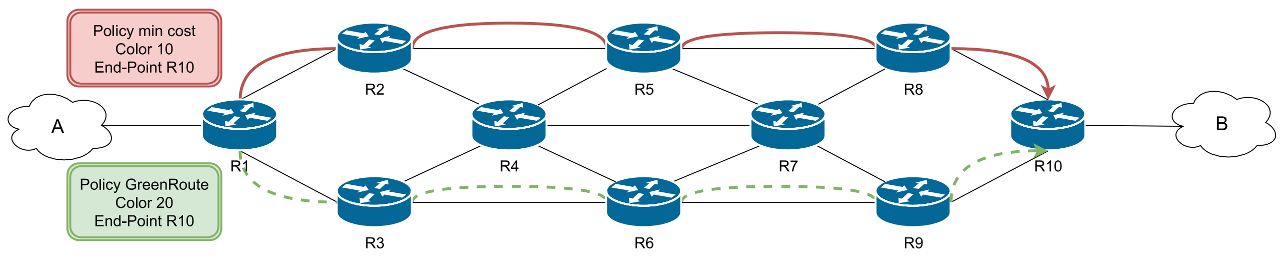


Figure 1.4: Example SR-Policy [10]

Explicit Path

From the many ways to steer traffic, the simplest one is also the most crucial in our thesis. We are talking about explicit path routing. On the headend, a configuration is applied that specifies which explicit Segments should be applied to a packet. As a result, the router includes this information in the corresponding packets. This method has the advantage of allowing an externally calculated path to be configured on the headend. At this point, the router is unaware of the policy's intent. It simply executes the instructions associated with it.

To ensure that only the greenest path is traversed, each node must be declared explicitly. This can be seen in Figure 1.5. At the ingress node all the SIDs are added to route the packet from ingress to egress. This leaves no wiggle room and it is therefore guaranteed that the path is the one previously calculated. This way we can give a reliable value for the environmental impact of this route.

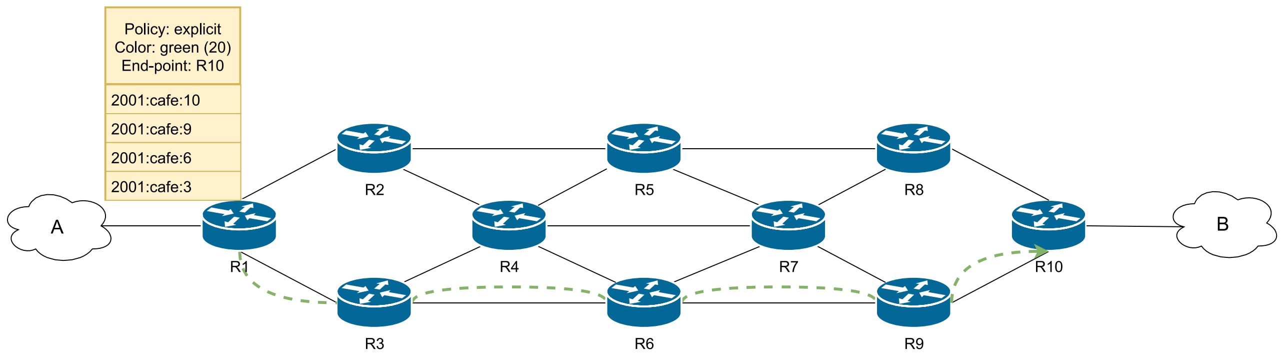


Figure 1.5: Example explicit path [10]

1.3.2 SR-Apps

The Institute for Network Solutions began developing applications that make use of the new possibilities that segment routing opens up shortly after the first draft of segment routing was released. The majority of these new applications are based on the idea that you can change the path packets take based on network metrics or characteristics. Another factor in common is the requirement that all apps must be cloud-native and scalable in a Kubernetes cluster.

1.3.3 Telemetry data

Our app heavily relies on telemetry data collection and utilization. In our case, we focus on Cisco ASR 9000 routers with IOS-XR software, which makes selecting the required resources very easy. The concept used by Cisco for sensor data selection is known as yang models [12], and it is a descriptive way to group sensor data and request it over a path. If the path is resolved, it sends one or more groups of sensor data in a predefined time interval.

As an example, we would like to receive data on power consumption. The path **Cisco-IOS-XR-sysadmin-asr9k-envmon-ui** is required for the ASR 9000 router with version 7.5.1. This path contains not only power data, but also sensor groups which we don't require. As a result, we specify the container that we want, which is a different name for group in the yang world. Other containers can be contained within a container. In our case, we want the container **oper** contained within the container environment to obtain our power data. **Cisco-IOS-XRsysadmin-asr9k-envmon-ui:environment/oper** is the entire path we need to configure. The router now understands what we want and begins delivering the configured telemetry data to the specified network address. There the data can be collected and stored in a time series database optimized for telemetry data.

1.3.4 Ecological aspects of datacenters

During our research we looked into the various measurements one can apply to rate the ecological aspect of a datacenter.

Power Usage Effectiveness (PUE)

> Data center operators use power usage effectiveness (PUE) to measure efficiency. A PUE value of 2.0 means that for every watt used to power IT equipment, another watt must be used to cool the IT equipment and distribute power. The closer the PUE approaches a value of 1.0, the more energy is spent on the computing activity itself. [13]

In Figure 1.6 you can see all the losses Google takes into account to calculate its PUE. Interestingly, only the green boxes are the IT components, while the red boxes are all counted as overhead needed to run the IT components. With all this, Google achieves a PUE of 1.10 which is quite good. [13]

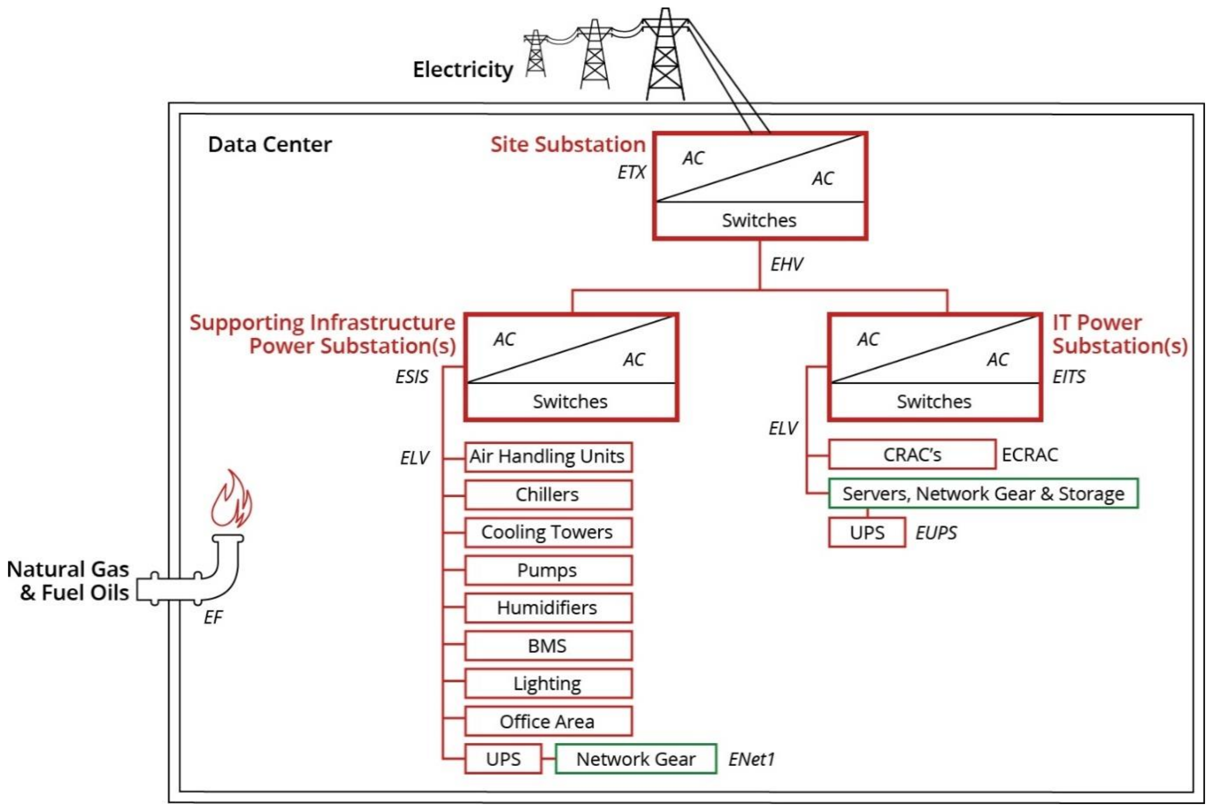


Figure 1.6: PUE measurement points of Google [13]

$$\text{PUE} = \frac{\text{ESIS} + \text{EITS} + \text{ETX} + \text{EHV} + \text{ELV} + \text{EF}}{\text{EITS} - \text{ECRAC} - \text{EUPS} - \text{ELV} + \text{ENet1}}$$

Figure 1.7: PUE formula of Google [13]

In Figure 1.7 you see the formula Google uses to determine their PUE based on the different measurements of Figure .

Carbone Usage Effectiveness (CUE)

> “Carbon usage effectiveness (CUE) is a metric for measuring the carbon gas a data center emits on a daily basis. The metric was developed by the non-profit consortium, The Green Grid.” [14]

The CUE can be calculated by taking the amount of electricity a datacenter consumes and multiply it with the local carbon factor, which is dependent on the power mix of the local electricity provider. The best score is 0.0 if you only receive power from renewable sources.

Water Usage Effectiveness (WUE)

> The Green Grid defines a sustainability metric as "the amount of water used on-site for data center operations, including humidification and on-site evaporation for cooling or energy production." WUE is calculated by dividing 'annual water usage' by

'IT computing equipment energy consumption.' WUE is measured in liters per kilowatt-hour (L/kWh). [15]

One aspect that may have been overlooked for a while was the amount of water tech companies consume to cool their carbon neutral datacenters located in the desert. A 15-megawatt datacenter can use up to 1.6 million liters of water per day. [15] So it makes only sense to also include this metric into the whole green path calculation.

1.3.5 Jalapeño

We discussed telemetry data in the previous chapter, which is sent to a specific address where it is collected, processed, and stored. Cisco's Jalapeño software is one solution for this. It is made up of an InfluxDB time series database as well as a Graph database, for storing topology data, called ArangoDB.

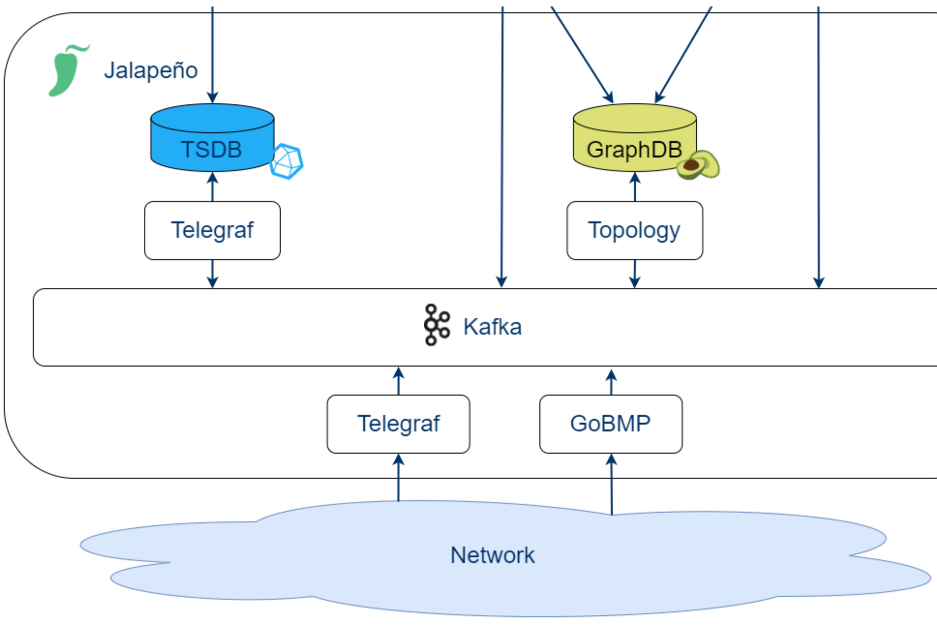


Figure 1.8: Architecture Jalapeño [16]

Data is delivered from the network to two processors, as shown in Figure 1.8: Architecture Jalapeño [16]. Telegraf is used for telemetry data, and a Cisco plugin is used to decode the data received from the routers. Jalapeño processes the received data for topology data using GoBMP. Both processors send their output to Kafka, which subsequently generates topics to which consumers can subscribe. One of the consumers is another Telegraf processor that prepares the data for InfluxDB. The second processor, the one consuming data from Kafka, is the Topology processor, which does the same thing for ArangoDB.

1.3.6 Jalapeño API Gateway

Once the data has been stored in both databases, it has to be obtainable. To assist with this, the INS began work on the Jalapeño API Gateway, which provides a well-defined interface for SR-Apps to access the collected data. The API Gateway offers two primary services. One is the request service, which allows you to request data on demand. The other service is the subscription service, to which one can subscribe to in order to be notified of topology changes.

SR-Apps can then request time ranges from the time series database and be notified when the topology changes, allowing them to recalculate the path based on the new information.

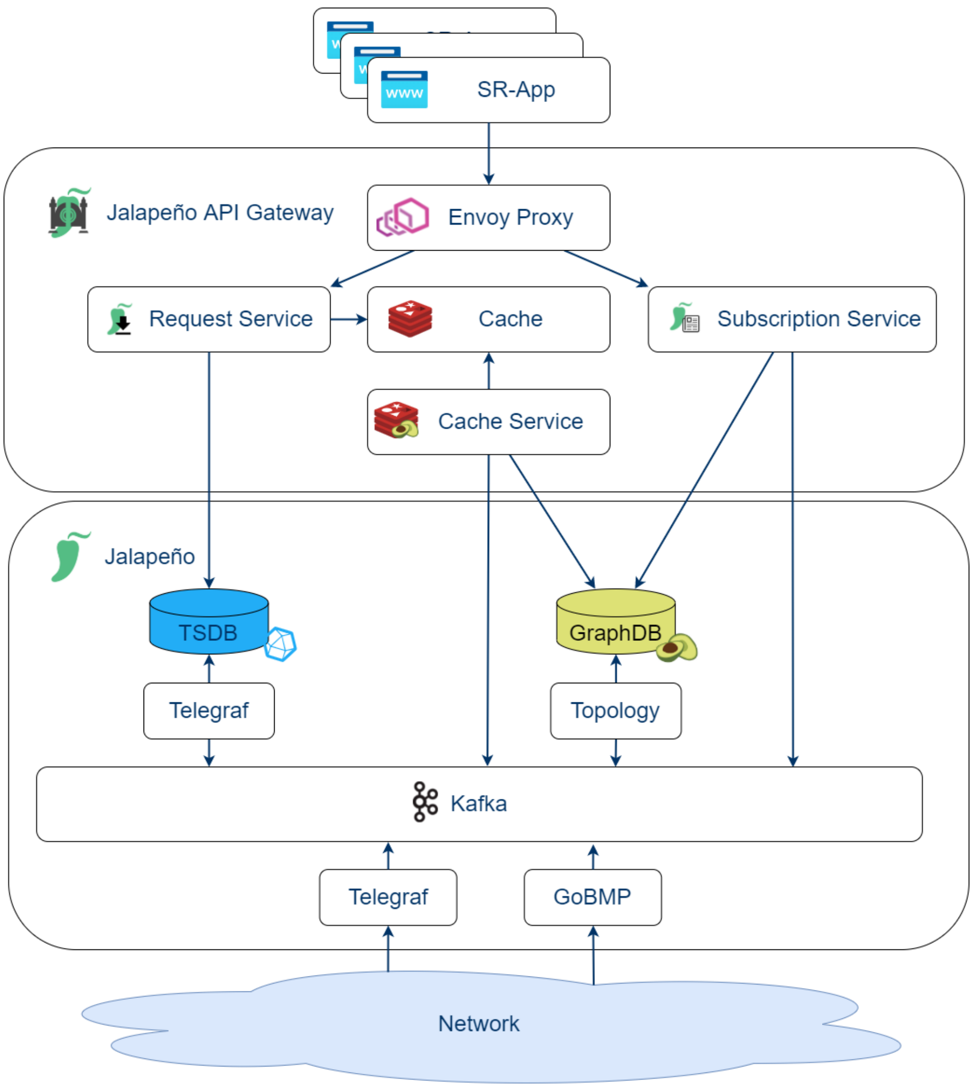


Figure 1.9: Architecture Jalapeño API Gateway and Jalapeño [16]

1.4 Goals and tasks

1.4.1 Problem

Every contribution counts in our world, in which we are fighting climate change. The network sector alone accounts for approximately 6% of total global CO_2 emissions. To be more precise, we're talking about the ever-expanding backbone networks that handle our daily traffic. With an annual CO_2 output of approximately 1.4Gt, it is worthwhile to investigate potential improvements [17]. One suggestion was to manage traffic over these networks using the newly developed capabilities of segment routing. This allows one to analyze the power consumption of routers in a network over time and then route traffic through the routers that require the least amount of electrical energy in relation to their throughput. In general, the newer the router the more efficient it is. When a router from a decade ago is compared to a new router from today, the consumed power per Gbps is reduced by 95% [18]. Keeping this in mind, it should be possible to identify the most efficient router per Gbps and route vast quantities of traffic through these routers. In the best-case scenario, you would consider the source of the energy, prioritizing routers powered by renewable energy.

1.4.2 Solution

In a first step we need to define what a green router is and how we can measure that. We need to gather useful metrics and figure out a system to represent a node with a single green score. We need to read the possible sensor data from the router and gather the other metrics by hand.

The second step is to design an algorithm which takes the router ratings and applies a K-shortest paths algorithm on it. This results in one or more green routes with a cumulated green score, which allows us to compare different paths for their environmental impact.

The green route can then be deployed. For this a SR-Policy needs to be created and a way needs to be found to configure the devices in the network. The SID list will then be configured on the ingress node and the packets should follow the calculated path.

To present the results in a meaningful way, a frontend needs to be designed which can display the network, calculate new green routes and display the results. To compare them to the fastest path, a way to compare both paths should be implemented.

To achieve an easy access for the INS, the application will then be deployed on the Kubernetes cluster of the INS.

2. Results and Discussion

The outcomes of our thesis are presented in this chapter. We showcase what we achieved while also providing information on the hidden features we implemented. We describe our successes in the chapter Accomplishments. Additionally, we describe how the outcomes have been achieved in the chapter on implementation.

2.1 Accomplishments

The backend API from the term project was significantly improved and expanded in this thesis. The underlying green index was expanded using various green metrics and thus improves the significance of how thorough a route actually is. In addition to the backend, a frontend was developed to illustrate the results from the backend for demonstration purposes.
All of the defined use cases were accomplished, as well as one additional use case.

2.1.1 CRUD greenest route

By using the HTTP POST method on the green routes API endpoint, a user can request the calculation of a new green route. To begin a new computation, they must provide both the ingress and egress node key. They in turn receive a JSON object containing the green route which includes the overall green score, each link that makes up the route, a segment identifier list consisting of all segments (as well as their SID's) and a defined SR-Policy for the deployment. A query parameter can be used to specify whether the SR policy should be configured directly in the network. The same is possible retroactively by deleting a green route, through which all associated data is deleted, and the route is deprovisioned completely from the network. In addition, the user has the option of requesting all past computations for any given green route, all green routes collectively, or a single green route by either its id or its accompanying ingress or egress node, each of which is recognized by its own unique node key.

2.1.2 Calculate paths

During the inception phase, multiple variants were evaluated for the second version of the green route calculation. The goal was to calculate multiple routes who may be equally green. Basically, there were four different possibilities to solve this requirement.

1. Develop an implementation based on Yen's k-shortest paths ourselves
2. Use an existing go math library which has a proven and widely used Yen's k-shortest paths algorithm
3. Use an existing go library with a K-Star implementation with only a small community
4. Switch the database from MariaDB to ArangoDB and use the database graph function k shortest paths

The decision was made for the second variant. The reasons were that it is an already tested and performant implementation which saved valuable time. The other solutions were either more work or more risk to implement. The decision for Yen's k-shortest paths algorithm was made because it performed better than other algorithms for our use case.

As soon as the backend receives the request for a new calculation, it first synchronizes the latest telemetry data and processes it to get an accurate result of the currently most ecological route. Subsequently, the individual green scores per node are calculated on a weighted basis, which is then used as the link cost so that the Yen’s k-shortest path algorithm can calculate the best path or paths.

2.1.3 Define stable route

We included an environment variable that governs the interval in which telemetry data is acquired, with which the average power consumption together with the throughput value is computed. This prevents routes from changing too rapidly. The variable’s default value is set to 10 minutes. Spikes in power consumption will have less of an impact in this interval, since the node's long-term measurement will be the more interesting indicator.

In addition, a frequency can be defined in which the generated green routes are recalculated and optionally automatically redeployed to the network.

2.1.4 Define green index

With this thesis, we have revised the approach to the definition of the green index from the term project. In the process, we were able to work out many possible metrics, both for the routers themselves as well as for the environment of the routers in data centers. The possible metrics are weighed individually and added up to a resulting green score. The weighing was roughly estimated and therefore is only a proposal without extended research backing it. This green score has a value between 1 and 100, where lower means more ecological and therefore better. The green metrics listed below were be found, but only the first four of them are actually utilized in the Green SR app for simplicity. Though it is very easy to manually add new green metrics and specify values for them. Only the power consumption and data throughput of the routers are obtained as telemetry data from the devices. All other data is determined manually, since there is currently no possibility to obtain it dynamically, but also because in many cases the data does not exist yet.

Name	Description/comment	Units	Weighting
Throughput per watt	Ratio between data throughput and power consumption. The higher the ratio the better.	Throughput: Mbit/s Watts: watt	8
Not renewable power source	Percentage indication of how much electricity does not come from renewable sources.	Percent	6
PUE	A measure of how effectively a data center utilizes energy, precisely how much energy is consumed by the computing equipment, called Power Consumption Effectiveness (in comparison to cooling and other operational overhead that supports the data center). The resulting value is normalized to the green score scale.[19]	Points	6

CUE	The Green Grid created the Carbon Usage Effectiveness metric to quantify the carbon emissions-related sustainability of data centers. CUE is the ratio of the energy consumption of IT equipment to the total CO_2 emissions produced by all data center energy use. The resulting value is normalized to the green score scale.[14]	Points	6
Cooling system	How efficient the cooling system is and whether it uses sustainable resources to operate.	Points	4
Climate compensation	How high the climate compensation of a data center is. Classified according to the fixed values from the following scale. 1: highly positive 20: low positive 40: climate neutral 80: climate negative 100: highly climate negative	Points	3
Datacenter place / landscaping	Where the data center is geographically located. Also considering how good the connection to the necessary resources is. As an example: A data center in the desert most likely performs worse.	Points	3
Construction without sustainable materials	Percentage of unsustainable materials used in construction of the data center.	Percent	3
Sustainability of scalability	Possibility to expand or extend the data center in a resource-saving way.	Points	1
Emergency setup sustainability	How sustainable the whole system is for emergency operation during a power outage.	Points	2
Efficiency of logistics and work processes	Basic efficient logistics and work processes. Certain certifications already exist for this.	Points	2

Table 2.1: Green metrics for the green index [20]

2.1.5 Get structured data

We must transform the data into our own entity types after retrieving the topology and telemetry data from the Jalapeño API Gateway service. The generated information is saved in a database for further use. In this thesis, a live synchronization of the network data via JAGW was implemented in order to always have the latest network status ready. Therefore, it is not needed anymore to sync the whole network before each calculation request. Besides that, the synchronization on demand as implemented in the term project is still possible. Due to limitations of the current Jalapeño API gateway version, certain connection options between nodes are still synchronized on demand, but the logical links and node information are completely synchronized live.

2.1.6 Deploy SR-TE policy

After a green route has been successfully calculated, it is deployed in the network according to a defined SR policy. Unfortunately, the configuration could not be done as planned via gRPC with Yang Models, because a certain field for SRv6 is missing in the currently available Yang Models. The alternative solution now works classically via direct commands via SSH on the routers. This unfortunately has some disadvantages, but there was no other viable option.

It is possible to make the configuration directly after the calculation, or to skip it and request it later separately. The SR-Policies API endpoint allows to retrieve all policies generated so far, filter them by id or associated green route, deploy them and also delete them, including the deprovisioning on the routers.

2.1.7 View routes

A frontend was implemented to display the network as a graph and to highlight calculated green routes. The user interface is capable of displaying the nodes dynamically in user-friendly way, so that the view is well-structured and there is no unnecessary overlapping of the links between the nodes. The algorithm is very likely to get an optimal arranging of the nodes on the first try, but sometimes the result may be not useful, and the calculation has to be started again manually. As soon as a green route is marked, the green scores are displayed for all links of the path with the total average green score of the selected green route. It is also possible to highlight individual nodes and their connections by hovering over a node.
The user interface also allows the generation of new green routes via a pop-up window. Additionally, all available green metrics can be enabled or disabled individually.

2.1.8 Compare scenarios

In order to show a comprehensible comparison between the green and the fastest path, it is also possible to calculate the fastest route via the frontend. The fastest route can then be compared with the greenest route, either in a split view with one graph per path or a combined view with different colors for each individual route.

2.1.9 Gather statistics

For research purposes, having some statistical data over time of a node's or an entire network's power usage and throughput rate is useful. The introduction of the node power consumption and node throughput entities, which hold all the determined values of all nodes along with the timestamp of their retrieval and or calculation, solved this problem.

2.1.10 Miscellaneous

To enable the deployment of the frontend and backend applications with their services, a Kubernetes deployment configuration had to be created. Since a manual configuration through a Kubernetes configuration file is time-consuming and more difficult to maintain, it was decided to create a Helm chart instead. Helm helps to define, manage, install, and upgrade Kubernetes applications. In addition, there are already Helm charts templates in a registry. A MariaDB template chart was used to deploy the required database for the backend. The Helm chart definition of the entire deployment is stored in its own repository, allowing quick deployment anywhere.

In order to be able to test the backend API easily, the API client Insomnia was used as mentioned in part B of the documentation. In order for everyone involved to have easy access to all requests available from the SR-App, an additional repository was created, where the latest configuration of Insomnia is always available and synchronized.

As already noticed in the term project, the use of multistage docker builds would be useful to keep the size of the final docker images as small as possible and to perform the CI/CD pipeline process faster.
Over the course of this project, we successfully switched to multistage builds. The first stages work as before with all the required steps for building and quality checking, but in the final images small distributionless images are used, which only have a size of Mbyte in the low two-digit range.

2.2 Implementation

This chapter explains how the solutions were achieved. Important decisions and considerations made for the final product are described in detail.

2.2.1 Backend

The backend, which manages all requests, network synchronizations, and path calculations, is the most crucial part of the system. It delivers uniform and condensed responses for each situation and responsible for validating the inputs.

For the API, we continued to use the Gin Web Framework as in the term project. It worked flawlessly and delivers on its promises regarding performance. Additionally, the documentation is comprehensive measured by Go standards and the community is relatively large. [21]

The *controller* package defines each endpoint. All endpoints are registered in Gin's *api* package upon application startup so that it can handle incoming requests with its API router. Our backend offers the endpoints listed below.

The base endpoint delivers a welcome message along with the URL to the OpenAPI standard documentation with Swagger.

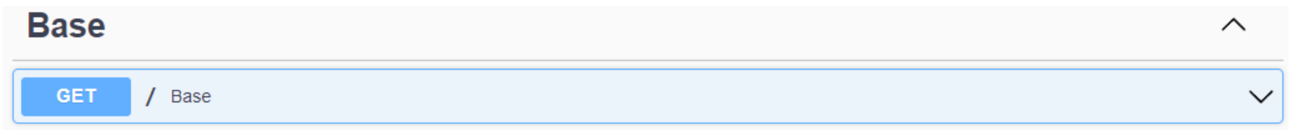
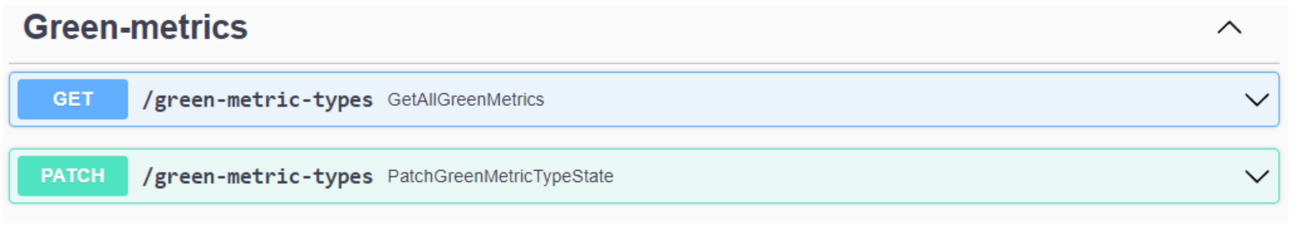


Figure 2.1: Endpoint base [10]

Using the following endpoints for the green metric types, the currently available ones can be listed and their activation status can be adjusted. The green route calculation algorithm always only uses the activated green metrics as a basis.

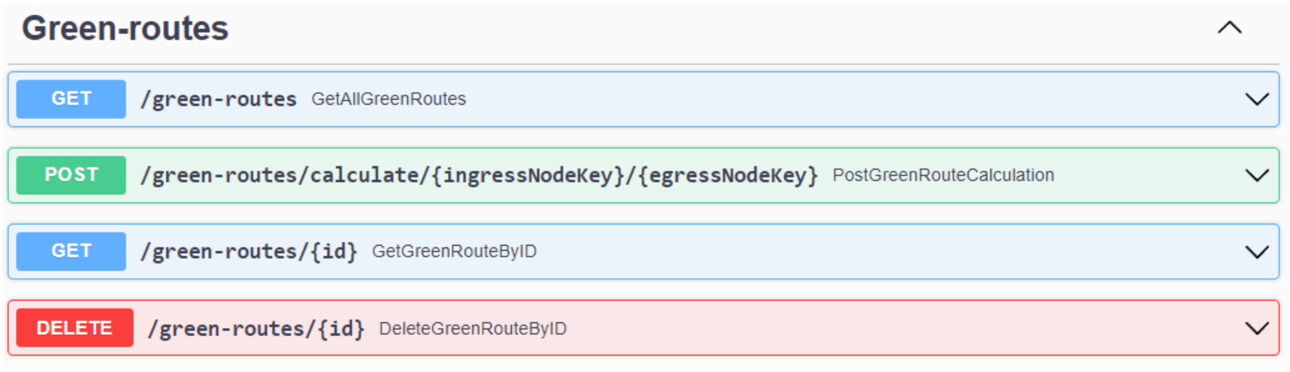

Green-metrics

GET /green-metric-types GetAllGreenMetrics

PATCH /green-metric-types PatchGreenMetricTypeState

Figure 2.2: Endpoint green-metrics-types [10]

By providing both the ingress node key and the egress node key, the green route endpoints enable the calculation of a new green route. After first requests have been made, it is possible to obtain the previously established green routes if any computations have been performed.
This has not changed compared to the term project, except that the SR-Policies are now additionally generated in the background and deployed to the network. This is managed by the respective separate endpoints more extensively. Furthermore, green routes can now be deleted via the API.


Green-routes

GET /green-routes GetAllGreenRoutes

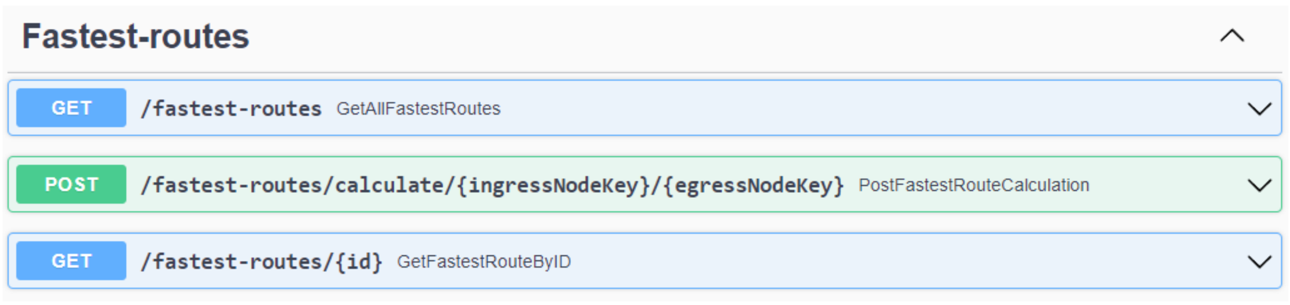
POST /green-routes/calculate/{ingressNodeKey}/{egressNodeKey} PostGreenRouteCalculation

GET /green-routes/{id} GetGreenRouteByID

DELETE /green-routes/{id} DeleteGreenRouteByID

Figure 2.3: Endpoint green-routes [10]

In addition to the calculation for the green routes, it is now possible to calculate the fastest traditional routes, for which the path is added up using the IGP link metric. This is required to compare the scenarios in the frontend, as described in chapter 2.2.4.


Fastest-routes

GET /fastest-routes GetAllFastestRoutes

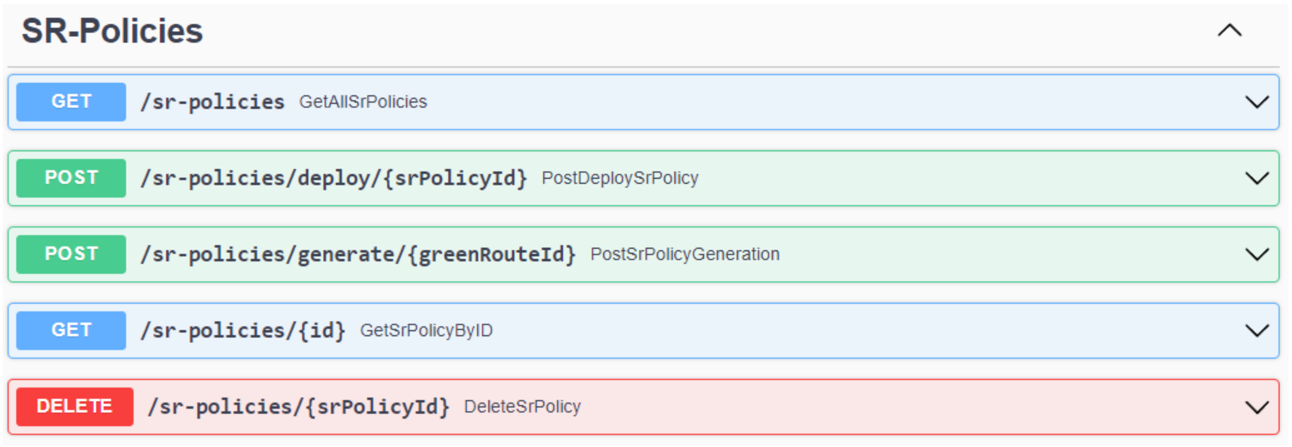
POST /fastest-routes/calculate/{ingressNodeKey}/{egressNodeKey} PostFastestRouteCalculation

GET /fastest-routes/{id} GetFastestRouteByID

Figure 2.4: Endpoint fastest-routes [10]

For each green route, SR policies are required, which are either created automatically after the calculation or generated and deployed separately via endpoints following in the Figure 2.5.
In addition, all previously created SR-Policies can be queried with filter options, e.g. based on the green

route or the id. A separate deletion of the SR-Policy only is also possible, which subsequently triggers the deprovisioning on the routers in the background.


SR-Policies

GET /sr-policies GetAllSrPolicies

POST /sr-policies/deploy/{srPolicyId} PostDeploySrPolicy

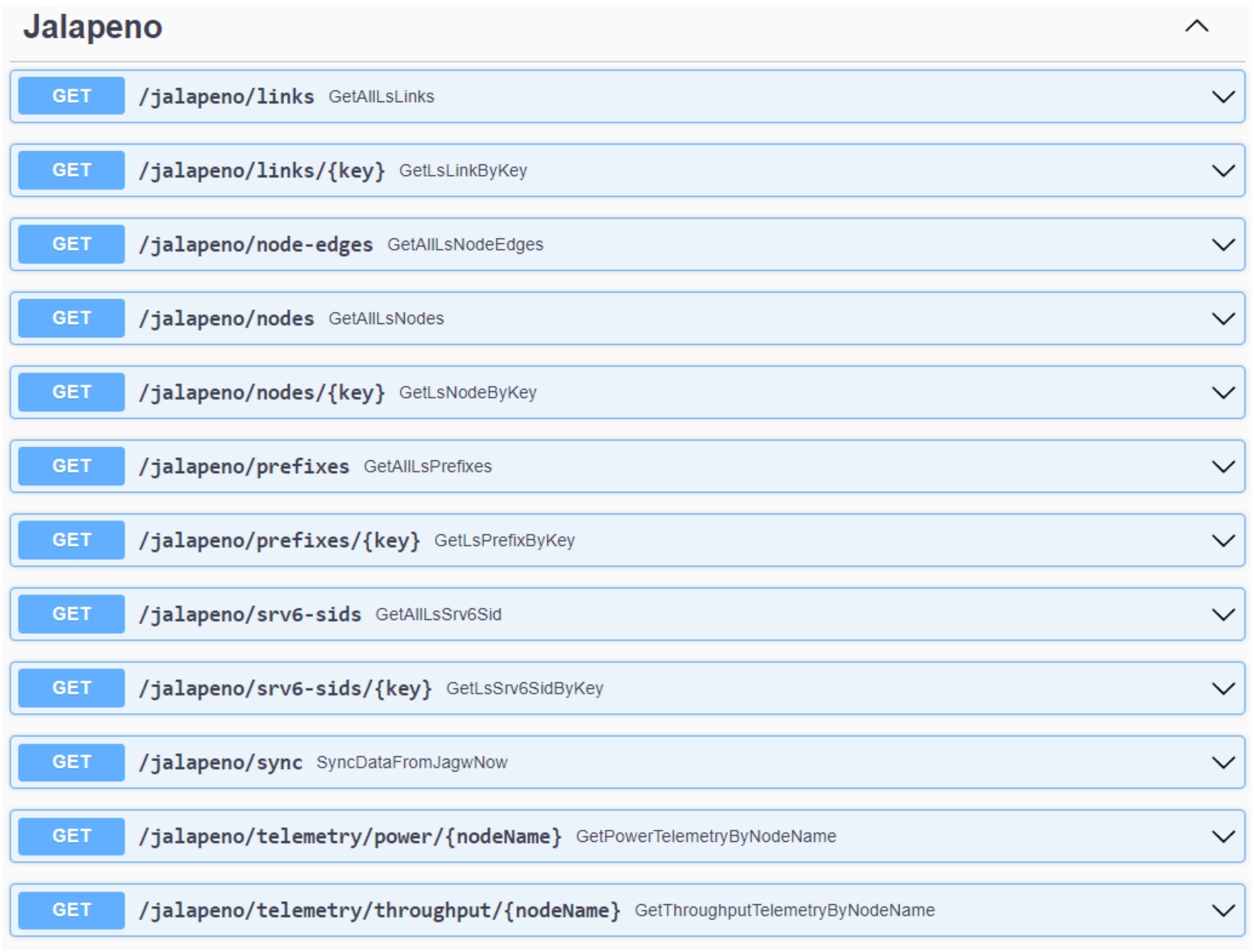
POST /sr-policies/generate/{greenRouteId} PostSrPolicyGeneration

GET /sr-policies/{id} GetSrPolicyByID

DELETE /sr-policies/{srPolicyId} DeleteSrPolicy

Figure 2.5: Endpoint SR-Policies [10]

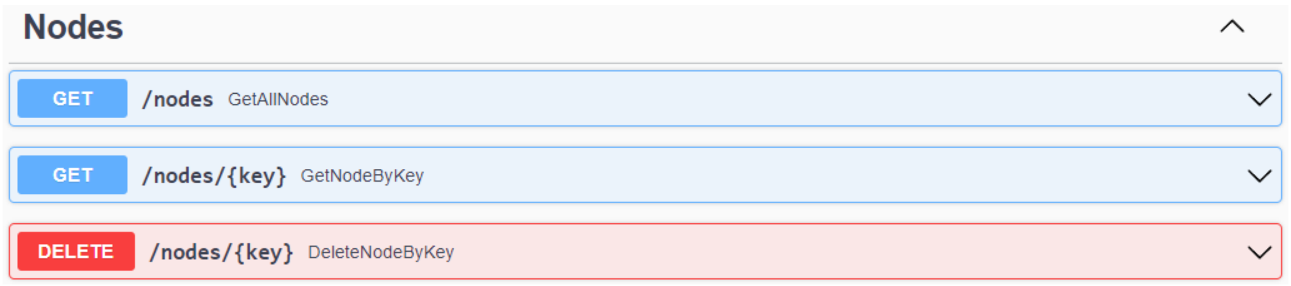
The Jalapeño endpoints offer a simple method for getting a list of all nodes, links, node-edges, prefixes or srv6-sids from the network that is connected. All data is obtained through JAGW, except for the prefixes, which have to be fetched directly from the Jalapeño database because JAGW does not provide them. A fresh synchronization of the topology data on demand can also be requested. Additionally, by supplying the name of the desired node, different telemetry data for that node can be requested.


Jalapeno

GET /jalapeno/links GetAllLsLinks
GET /jalapeno/links/{key} GetLsLinkByKey
GET /jalapeno/node-edges GetAllLsNodeEdges
GET /jalapeno/nodes GetAllLsNodes
GET /jalapeno/nodes/{key} GetLsNodeByKey
GET /jalapeno/prefixes GetAllLsPrefixes
GET /jalapeno/prefixes/{key} GetLsPrefixByKey
GET /jalapeno/srv6-sids GetAllLsSrv6Sid
GET /jalapeno/srv6-sids/{key} GetLsSrv6SidByKey
GET /jalapeno/sync SyncDataFromJagwNow
GET /jalapeno/telemetry/power/{nodeName} GetPowerTelemetryByNodeName
GET /jalapeno/telemetry/throughput/{nodeName} GetThroughputTelemetryByNodeName

Figure 2.6: Endpoint jalapeno [10]

The node endpoint is used for statistical analysis and gives the consumer access to a history table of the calculated power consumptions and synced data throughputs over time.
In this project, nodes can now be deleted to simulate a network change. Though this is only possible in the mocked data mode of the Green SR app. The delete request is otherwise rejected by the API.


Nodes

GET /nodes GetAllNodes
GET /nodes/{key} GetNodeByKey
DELETE /nodes/{key} DeleteNodeByKey

Figure 2.7: Endpoint nodes [10]

2.2.1.1 Architecture

The architecture we created was influenced by the two key factors-defined in the term project. Building a cloud-native application was a prerequisite and choosing to use the domain driven design paradigm was another.

The application has to be created using the Twelve-Factors methodology in order to be able to deploy it to the INS's Kubernetes cluster. [22] Due to this, it was decided to simply store the topology in the database in order to save complexity and avoid having to use a cache. Essentially, this means that all instances of the application share the same data without synchronization.

Compared to the term project, nothing was changed in the architecture, since the Domain Driven Design approach had proven itself and a change would have taken up unnecessary time.
The DDD approach also helped to maintain a clean project structure and to present the business logic in a comprehensible way. The separation of layers could be consistently maintained and helps the now again significantly grown software to remain maintainable.
Furthermore, the use of transfer objects and entities to define both the data sources and the database at the same time has proven itself to the end.

The repository layer is still completely replaceable in case of a future switch to another database service. The strategy of consistent layer separation and DDD has yet again proven itself here.

Furthermore, no changes were made to the architecture compared to the base project.

2.2.2 Calculation

Since it is part of the business logic, the greenest route is calculated in the service package. There is no need to sync the topology data on each request anymore since we have implemented a way to perform live updates via a subscription to network changes. This saves a lot of resources for the calculation and therefore the calculation itself also becomes much more performant, especially for very large networks.

2.2.3 Testing

Our project required a significant amount of testing, particularly automated integration tests. As already defined in the term project, we decided against test-driven development and kept that choice for this project.
We were again able to mock the repository layer as well as the service layer component for all new introduced code. As a result, we were able to achieve a test coverage of more than 80%, which is an improvement of around 15% in comparison to the term project.

2.2.4 Frontend

The frontend was implemented in the TypeScript programming language using the React framework, supported by the Material UI framework. Material UI, also called MUI, provides a grid layout system, prebuilt CSS classes and components that facilitate frontend development. The graphical user interface consists of several components, each of which has a specific task. The heart of it is the component displaying the network graph and routes. This component was implemented utilizing Sigma.js, with underlying the graphology library. Graphology handles the graph data model and the algorithms. Sigma.js itself takes care of the graph rendering and interactions. To be able to use Sigma.js better in a React environment, the library react-sigma v2 was used, which already provides various React components, hooks and contexts for the use of Sigma.js. [23, 24]

For all http requests the library Axios is used, which also handles all error handling based on the http status codes. Axios is a promise-based HTTP client for both the browser and Node.js. [25]

To compare the routes optimally, a combined view was implemented in which both paths can be superimposed. However, it is optionally possible to display the greenest and fastest route in a separate graph and compare them this way.

Figure 2.8 shows the complete frontend with the highlighted calculated green and fastest routes. The two buttons at the top can be used to request new routes. The 3 outlined buttons below are (from left to right) the button to switch from the combined to the single view of the routes, the button to open the pop-up to adjust the green metric activation status and on the far right a button to display a guide for the colors in the graph. At the bottom of the screen is the centerpiece: The network graph with the selected routes. In the upper right corner of the network graph, you can search for nodes and in the lower right corner there are additional options to customize the graph (e.g. full screen mode or changing the display algorithm).

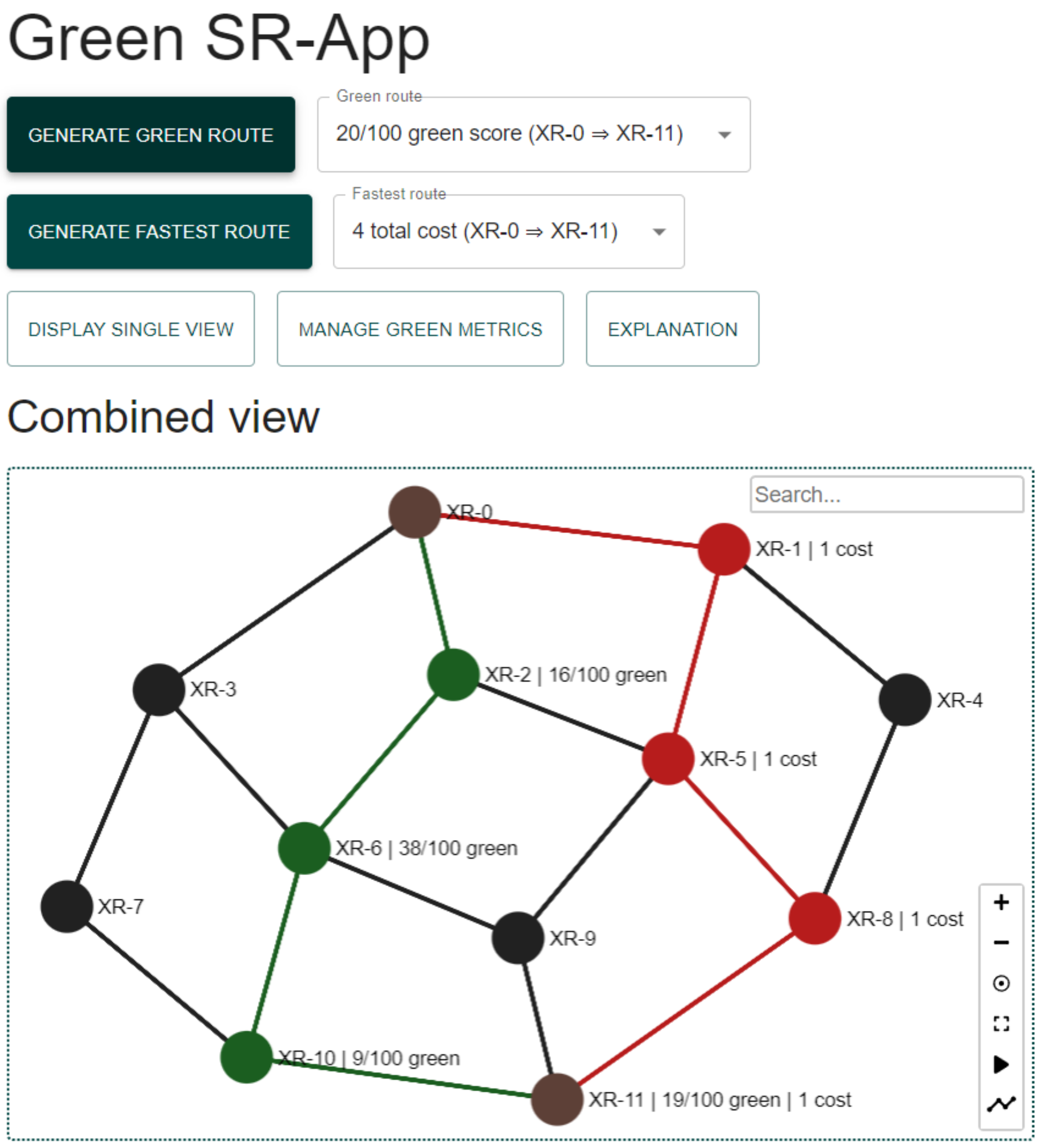


Figure 2.8: Green SR-App frontend [10]

To analyze the nodes with their connections more easily, you can hover over a node to highlight the surrounding links and nodes, as shown in Figure 2.9.

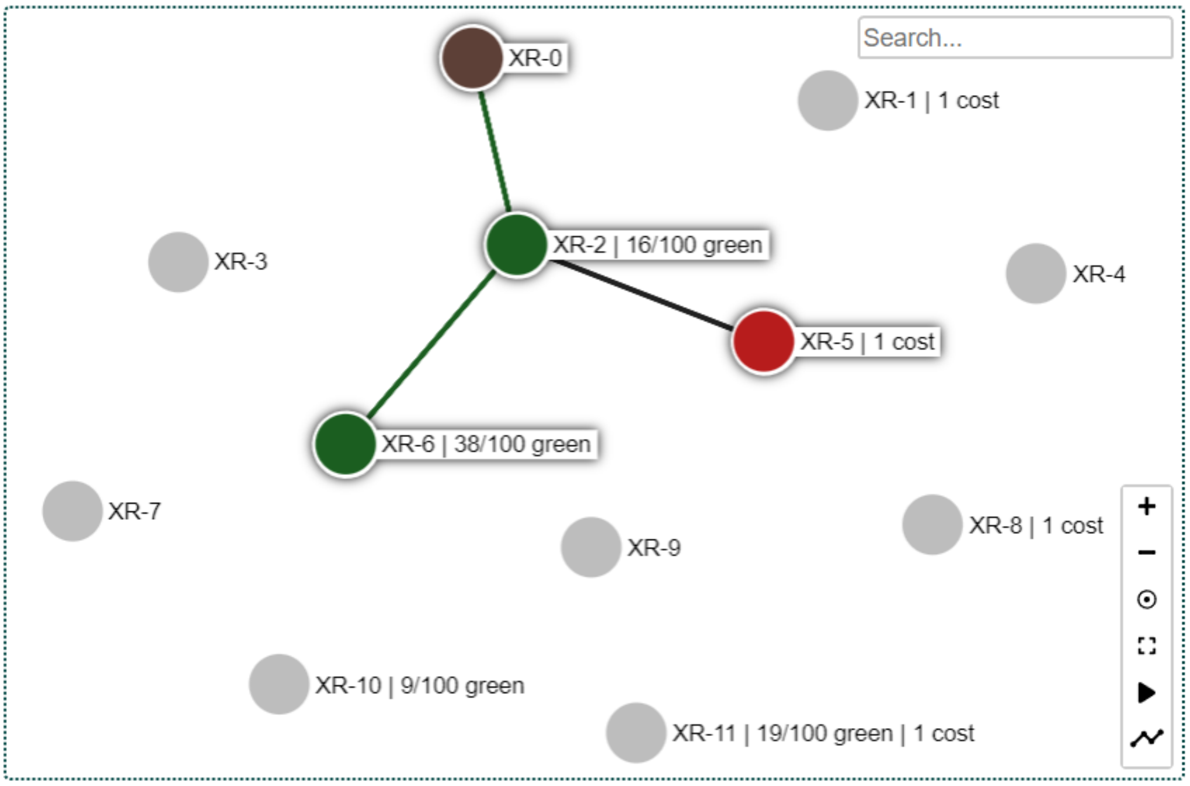


Figure 2.9: Green SR-App frontend – hovering over node [10]

3. Conclusion

In this chapter, we go over each use case and describe whether or not it was achieved and why. A blue color ✓ identifies the mandatory use cases. Additionally, the optional use cases are identified by a brown color ✓.

3.1 Requirements revisited

UC01: CRUD greenest route ✓

"As a User, I can get the greenest route over an API, so that I can lower my energy consumption."

This use case is fully satisfied. The user can calculate a new green route and retrieve or delete already calculated routes. No HTTP request matching the update in CRUD, since this is achieved by creating a new calculation with the same ingress and egress node, which subsequently updates the green route.
It is also possible to trigger a network data synchronization with Jalapeño API Gateway. In normal circumstances, however, this is not necessary because the network data is updated continuously via the subscription service.

UC02: Calculate paths ✓

"The Green SR-App can calculate the greenest (most energy efficient) route through a network."

This use case was already completed during the term project. Nevertheless, in this bachelor thesis, the underlying algorithm was replaced by Yen's k-shortest path algorithm, so that the requirement for calculating one or multiple routes at once can be met. In a network with 1000 nodes and many more links in between, the calculation finds a path in less than one second on average. A more than double as fast handling for calculation was achieved compared to the term project, since no full network synchronization has to be completed in advance. Only the telemetry data is synchronized before each request, since all other data is updated continuously via JAGW subscriptions.

UC03: Define stable route ✓

"As a User, I expect that the calculated most efficient path stays stable for the duration I define."

This use case was completed. As in the term project, a time period can be specified over which the telemetry data is obtained for which the average is calculated. Furthermore, the user can define an interval in which the calculated green routes should be renewed, including the subsequent automatic deployment in the network. The data required for the calculation is obtained continuously and only the only operation performed each time the interval elapses is to recalculate the path.

UC04: Define green index ✓

"As a User, I want to combine different metrics to a green index, which can be used to calculate the greenest path(s)."

This use case was ~~also~~ successfully completed. An unlimited number of metrics can be entered in the system and activated or deactivated at runtime depending on the requirements. For each calculation, the activated green metrics are either synchronized (telemetry data) or taken from a manually created data set, which is normalized and individually weighed to a total value. The telemetry-based metrics are automatically synchronized from JAGW while manually supplied data sets are auto-generated and then saved in the database. Currently, four such green metrics are used in the application. Extending the application with manual metrics is made easy for the operator.

UC05: Get structured data ✓

"The algorithm needs the necessary fields from the Jalapeño API Gateway to calculate the most efficient route."

Since UC02 would not have been possible without this use case, this one is implicitly fulfilled. The entire topology data may be synchronized on demand or via the live subscription, with the results being stored in a database. The synchronization runs entirely through JAWG, except for the information for the prefixes, which is obtained directly from ArangoDB from Jalapeño. To meet the use case's requirements, the latest telemetry data is completely synchronized before every green route calculation to have the latest available data ready for calculations.

UC06: Deploy SR-TE Policy ✓

"As a User, I can deploy the generated SR-TE Policy on the routers to steer traffic according to the defined rules."

UC2 was fully met, but not entirely as planned. With each calculation, the SR policy is automatically generated and then the network is configured for the green route. These steps can also be disabled via a request of the calculation and then triggered manually. After configuration, the network uses the new green route. The configuration is correct on the ingress and the egress router, but it could not be tested whether this route is indeed being used now.
Unfortunately, it was not possible to configure the routers as planned with the help of Yang Models via gRPC, as already outlined previously. Instead, the router configuration is done with direct commands to the router via SSH.

UC07: View routes ✓

"As a User, I want a simple web interface to see the chosen path(s) and be able to recalculate the routes."

UC07 is fully completed. A frontend has been realized, which pulls data from the backend to dynamically display the complete network graph. Green routes can be requested via the frontend and are subsequently highlighted in the displayed network graph. All previously created green routes can be selected via a dropdown.
Additionally, it is possible to display the green metrics and adjust their activation status.

UC08: Compare scenarios ✓

"As a User, I want to demonstrate the power savings of the greenest route(s) compared to the fastest route(s)."

This use case has been completed. In addition to calculating green routes, it is also possible to calculate fastest routes via the frontend. They are used to compare traditional routes with the new green routes as well as to demonstrate the possibilities of segment routing. For the highlighted routes, the green scores and IGP link metrics per node are displayed to give the user an idea of how the routes perform in general. In addition, the total and cumulative values for both routes are displayed in the dropdown selection.

UC09: Gather statistics ✓

"As a User, I want historical statistics to understand the system and how it behaves."

This additional use case was completed. The application keeps track of each node's calculated metrics after each calculation. Additionally, each synchronization stores an entry in the history with the power consumption and data throughput values from each node. This makes it possible to track how this metrics changes over time.

3.2 Learnings

In this chapter, we outline our experiences and discuss the things we would change if we could go back in time.

In order to develop more efficiently and not to reach a dead end in the implementation, it would be better in the future if the functionality of the JAGW is tested more extensively in the elaboration phase. In many cases, strategies for implementation had to be adapted because the API gateway unexpectedly returned errors, of which some were more serious.

In general, the bachelor thesis taught us once again that software parallelization is a double-edged sword. Always parallelizing everything as much as possible does not necessarily mean that it is the best solution. It is always important to validate which parallelization factor is optimal. In addition, a second very important point is to use fully synchronized data structures where needed to avoid data inconsistencies, which are usually very difficult to detect and analyze. Fortunately, Go natively offers a very good parallelization capability and provides the necessary synchronized data types.

As both team members are used to invest a lot in the quality of a software application, it was still sometimes difficult for us to focus more on new features rather than further improving the quality of the existing ones. This held us back from time to time and slowed down development to some extent. In the end, however, we think that we have found a healthy balance between introducing new features and the enhancement of quality.

In addition to the learnings mentioned, it is noticeable that we as a team have previously learned a lot from the term project, which is why there were basically no further measurable learnings.

3.3 Next steps

In this chapter, we'll go through the aspects of the implementation that could be improved. In the forecast, we'll look at the features we wish to add or improve upon, as well as provide some insight into the long-term plans.

3.3.1 Improvements

There is still a modest number of things to improve in the application. Some are dependent on external Services and cannot be fixed in the application itself. Others require a refactoring or rethinking of the code structure, for which there was not enough time during the project.

Jalapeño API Gateway / Jalapeño

One external task is to implement the subscription service of JAGW not only for nodes and links, but also for the important node edges needed to link the other two collections. For now, it is only possible to get this data on demand.

The data received from JAGW is not acceptable in its original form. Various keys and IDs do not match and there are many inconsistencies, which makes everything more complex. This led to the consideration of many special cases only to handle the data inconsistencies. If these problems were fixed at the source, a lot of logic could be omitted in the application.

Currently, if the application runs in the subscription mode, the connection is terminated after five minutes by JAWG. This, in its current form, is not an acceptable behavior. There is no easy to implement and maintainable way through which this problem can be fixed on the client side.

In the current release the application directly accesses the ArangoDB in Jalapeño to get data from the l3vpn_v4_prefix table. It would be much cleaner if this table would also be included into JAGW and therefore be accessible via its subscription/request service.

When the application requests all active nodes from the request service, it receives not only the existing nodes but also the deleted ones. This is another case where a special handling had to be included to handle this scenario.

Network configuration

A second part which needs a change from outside of the application is the fact that the newest yang models for traffic engineering lack the SRv6 capabilities and only offer MPLS. As long as this field is missing it is not possible to improve the network configuration. It would be cleaner if the whole configuration process could be handled per gRPC and the Yang models. This way one could generate the data structure for Go and would have established processes. This would implicitly improve the time to configurate the devices massively since only one message with alle the information would have to be sent, instead of the multiple SSH commands which need to wait for a response from the router.

Router changes handling

In the current version, SR-Policies are deployed to the network, but changes cannot be acted upon because the Green SR-App does not receive any information about them. If, for example, a configuration is adjusted manually on a router, the backend does not know this change and therefore has an inconsistent status in relation to the connected network.
In addition, configured green routes can be deprovisioned, but the router loses the configuration where it should route traffic for the deleted path. Here it would be necessary to save a backup of the current path routing settings before provisioning the green routes, so that this can be restored when deprovisioning.

Multiple paths

Right now, it is possible to calculate multiple best paths. But the new won possibilities are not yet used. One potential future use case would be to take the next best path in case the currently used path goes down, e.g. because a node has gone offline. This way there would be a smooth transition without the need of calculating a new path.

Strict separation of service and repository layer

Over the whole development time, the line between service layer and repository layer has become very blurry for some services. For longevity it would be beneficial to prune the service packages of repository tasks and the repository packages of service tasks. In the same step it would also be useful if all services had the same processes for the same tasks with a set of standardized functions in each service who have the same responsibilities. Only those standard functions would be allowed to access the repository layer.

3.3.2 Innovations

There are not only possible improvements but also new ideas which could be employed to improve the user experience even more.

The first step would be to increase the number of metrics available to the calculation. Possible candidates can be seen in Table 2.1. With a broader spectrum of metrics, it is possible to create a better balanced environment for the green operators and punish the bad ones more. So the result in the end will be even more accurate.

One upgrade would be to analyze the paths for equally green paths with different routes. If some sections could be parallelized it would spread the traffic over multiple nodes and implicitly also increase stability by preventing new calculations in case a node goes down.

To give the user a better understanding what routes are currently deployed on the network it would be a good feature to read the traffic engineering configuration from router. This way the application could provide information for each router on the frontend, including their configured routes as well as some other statistics such as the values of the metrics used in the calculation.

Another feature on the frontend would be the possibility to display certain statistics. For example, a graph plotting the power consumption of certain nodes, green routes or the whole network over the last 24 hours, the last week, and the last month. The same could also be done for the throughput in absolute values as well as in percent of their max capacity. This way the user can get a better understanding on the behavior of the network and where to potentially achieve the biggest improvements.

List of Figures

List of Tables

Glossary

ArangoDB	ArangoDB is a free and native open-source database system with multiple models. It supports three data models (key/value, documents, and graphs) with a database core and a unified query language called AQL (ArangoDB Query Language).
Data transfer object	A data transfer object (DTO) is an object that carries data between processes and application layers.
Domain driven design	Domain driven design is an approach to modelling complex software. The modelling of the software is significantly influenced by the technicalities to be implemented in the application domain.
Envoy Proxy	Envoy is an open-source edge and service proxy, designed for cloud-native applications.
Gin	Gin Web Framework is an API framework for Go.
GoBMP	Is basically an implementation of Open BMP (RFC 7854) protocol's collector in Golang.
Green index	The green index contains all green scores and ranks all routers by their ecological aspects. It is the measurement data basis for the green route calculation.
Green score	A green score is a numeric value between 1 and 100, where 1 is better and 100 worse in sense of how ecological something is. It is merged by multiple metrics which measure ecological aspects.
gRPC	gRPC is a modern open-source high performance Remote Procedure Call (RPC) framework that can run in different environments. It can efficiently connect services for multiple purposes.
Helm	Helm charts help define, install, and upgrade Kubernetes applications and is basically a package manager for Kubernetes.
IGP link metric	Interior Gateway Protocol link metric is an edge/path cost value for a link from one node to another.
Interior Gateway Protocol	An autonomic system's internal network information exchange protocol type.
Jalapeño	System developed by Cisco, which collects and processes data from attached networks, including telemetry data.
Kafka	Kafka, developed by Apache, is an open-source distributed event streaming platform used for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

Kubernetes	Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.
Metric	A standard of measurement.
OpenAPI specification	Defines a standard interface to RESTful APIs for both humans and computers
React	React is a JavaScript software library that provides a basic framework for outputting user interface components of web pages.
Redis	Is an open source, in-memory data structure store, used as a database, cache and message broker.
Request for comments	Request for comments documents contain technical specifications and organizational notes for the Internet. RFCs produced by the Internet Engineering Task Force (IETF) cover many aspects of computer networking.
Yen’s k-shortest path	The shortest routes between two nodes are calculated using Yen's Shortest Path technique. The algorithm supports weighted graphs with positive relationship weights.

Acronyms

DDD	Domain driven design Glossary: Domain driven design
DTO	Data transfer object Glossary: Data transfer object
Go	Golang Glossary: Golang
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol Glossary: Interior Gateway Protocol
INS	Institute for Networked Solutions at the Eastern Switzerland University of Applied Sciences
JAGW	Jalapeño API Gateway developed by the INS
MPLS	Multi-Protocol Label Switching
RFC	Request for comments Glossary: Request for comments
SID	Segment ID
SR	Segment Routing
SR-App	Segment Routing Application
SRv6	Segment Routing with IPv6 Addresses

References

[1] J. Hauser and P. Schlumpf, “Term project Green Routing: Autumn Term 2021,” Term project, Institute for network solutions, OST – University of Applied Sciences, Rapperswil, 2021.

[2] F. Dabaghi, Z. Movahedi, and R. Langar, “A survey on green routing protocols using sleep-scheduling in wired networks,” *Journal of Network and Computer Applications*, vol. 77, pp. 106–122, 2017, doi: 10.1016/j.jnca.2016.10.005.

[3] K. S. Ghuman and A. Nayak, “Per-packet based energy aware segment routing approach for Data Center Networks with SDN,” in *2017 24th International Conference on Telecommunications (ICT)*, Limassol, Cyprus, 2017, pp. 1–6.

[4] B. Balakiruthiga, P. Deepalakshmi, S. N. Mohanty, D. Gupta, P. Pavan Kumar, and K. Shankar, “Segment routing based energy aware routing for software defined data center,” *Cognitive Systems Research*, vol. 64, pp. 146–163, 2020, doi: 10.1016/j.cogsys.2020.08.009.

[5] R. Carpa, O. Gluck, and L. Lefevre, “Segment routing based traffic engineering for energy efficient backbone networks,” in *2014 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, New Delhi, India, 2014, pp. 1–6.

[6] S.-K. Jo, “TOWARD A GREENER INTERNET - Design and evaluation of green IP and content routing for sustainable communication networks,” UNSPECIFIED, 2021.

[7] *Segment Routing Architecture*, IETF RFC8402, Internet Engineering Task Force (IETF). [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8402

[8] *Segment Routing over IPv6 (SRv6) Network Programming*, IETF RFC8986, Internet Engineering Task Force (IETF). [Online]. Available: https: //datatracker.ietf.org/doc/html/rfc8986

[9] *Segment Routing Policy Architecture draft-ietf-spring-segment-routing-policy-22,* Internet Engineering Task Force (IETF). [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-spring-segment-routing-policy-22

[10] J. Hauser and P. Schlumpf, “Bachelor thesis Green Routing: Spring Term 2021,” Bachelor thesis, Institute for network solutions, OST – University of Applied Sciences, Rapperswil, 2022.

[11] *IPv6 Segment Routing Header (SRH)*, IETF RFC8754. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8754

[12] YangModels, *YangModels.* [Online]. Available: https://github.com/YangModels/yang (accessed: Dec. 21 2021).

[13] Google, *Efficiency.* [Online]. Available: https://www.google.com/about/datacenters/efficiency/#measuring-efficiency

[14] E. Watkins, *Carbon usage effectiveness (CUE).* [Online]. Available: https://www.techtarget.com/searchdatacenter/definition/carbon-usage-effectiveness-CUE

[15] R. Miller, *Data Center Water Use Moves to the Forefront.* [Online]. Available: https://www.datacenterknowledge.com/archives/2012/08/14/data-center-water-use-moves-to-center-stage

[16] M. Bongard, *Jalapeño API Gateway: A simple, light-weight, cloud-native API Gateway for Jalapeño.* [Online]. Available: https://jalapeno-api-gateway.github.io/jagw-docs (accessed: Dec. 21 2022).

[17] Global e-Sustainability Initiative, *Smart 2020: Enabling the low carbon economy in the information age.* [Online]. Available: https://gesi.org/research/download/7 (accessed: Jun. 22 2022).

[18] Cisco SP Routing Team, *Optimize Power Consumption.* [Online]. Available: https://xrdocs.io/asr9k/blogs/2018-09-06-power (accessed: Dec. 21 2021).

[19] The Green Grid, *PUE: A COMPREHENSIVE EXAMINATION OF THE METRIC.* [Online]. Available: https://www.thegreengrid.org/en/resources/library-and-tools/20-PUE:-A-Comprehensive-Examination-of-the-Metric

[20] U. Weibel, *Ökologie von Swisscom Rechenzentren*. Interview mit einem Swisscom Mitarbeiter. Online.

[21] Gin Team, *Gin Web Framework.* [Online]. Available: https://gin-gonic.com/ (accessed: Jun. 22 2022).

[22] A. Wiggins, *The Twelve-Factor App.* [Online]. Available: https://12factor.net/ (accessed: Sep. 29 2021).

[23] OuestWare, *React Sigma: A list of react components to display graph with sigma.js.* [Online]. Available: https://sim51.github.io/react-sigma/ (accessed: Jun. 23 2022).

[24] Sciences-Po médialab and OuestWare, *Sigma.js: a JavaScript library aimed at visualizing graphs of thousands of nodes and edges.* [Online]. Available: https://www.sigmajs.org/ (accessed: Jun. 23 2022).

[25] M. Zabriskie, *Axios: Promise based HTTP client for the browser and node.js.* [Online]. Available: https://axios-http.com/ (accessed: Jun. 23 2022).

B. Project Documentation

Change history

Version	Date	Changes	Responsible
1.0	06.03.2022	Finished initial project plan (chapter 2)	Jonas H. and Pascal S.
1.1	13.03.2022	Finished requirements specification (chapter 1) and make small construction milestones changes in the project plan.	Jonas H. and Pascal S.
1.2	20.03.2022	Finished development concept and most parts of architecture and design specifications.	Jonas H. and Pascal S.
1.3	29.03.2022	Revalidated and updated risks and risk graph. Risk Ri10, Ri12 and Ri4 are considered as occurred.	Pascal S.
1.4	17.04.2022	Risks Ri9 have been eliminated or can no longer occur.	Jonas H.
1.5	02.05.2022	Risk Ri1 is considered as occurred.	Jonas H.
1.6	17.05.2022	Updated timetable, milestone dates and important dates due to project end postponement.	Jonas H.
1.7	21.06.2022	Update development concept, backend architecture, Domain Model and 12-Factor Methodology. Create API documentation, entity relationship diagram, SonarQube quality gate status parts and small other parts.	Jonas H. and Pascal S.
1.8	22.06.2022	Final small changes due to proofreading.	Jonas H. and Pascal S.

Contents

1. Requirement specification

1.1 Use Cases

This chapter covers the requirement specification defined during the project's inception and early elaboration phases. The functional requirements were defined in the form of use cases and user stories. Non-functional requirements are addressed further in the chapter.

We divided the use cases and user stories into mandatory and additional ones.

● The purple color marks the use cases which are expected to be developed.

● The pink color marks the additional use cases.

1.1.1 Actors

In Table 1.1: Use case actors we listed all the different actors and their role in the context of our application. An actor can be a human or an additional software system which interacts with the system.

Actor	Description
User	A user can read, start new calculations and deploy new SR-TE policies. It is assumed that the user is already authorized for access since they are on a secure network.
Jalapeño API Gateway	The API Gateway is the provider of parts of the data the application needs. It can notify the subscribed services whenever the topology has changed.
Jalapeño	Jalapeño is the data collector system from the network and used by the API Gateway as data source.

Table 1.1: Use case actors

1.1.2 Use Case diagram

The use case diagram in Figure 1.1: Use case diagram [1] gives an overview over the relation between the different use cases. An include indicates, that a previous use case must be fulfilled before the next use case can be accomplished. Whereas an extend indicates an optional extension of the existing use case, whereby the latter doesn't depend on the extension to perform its task.

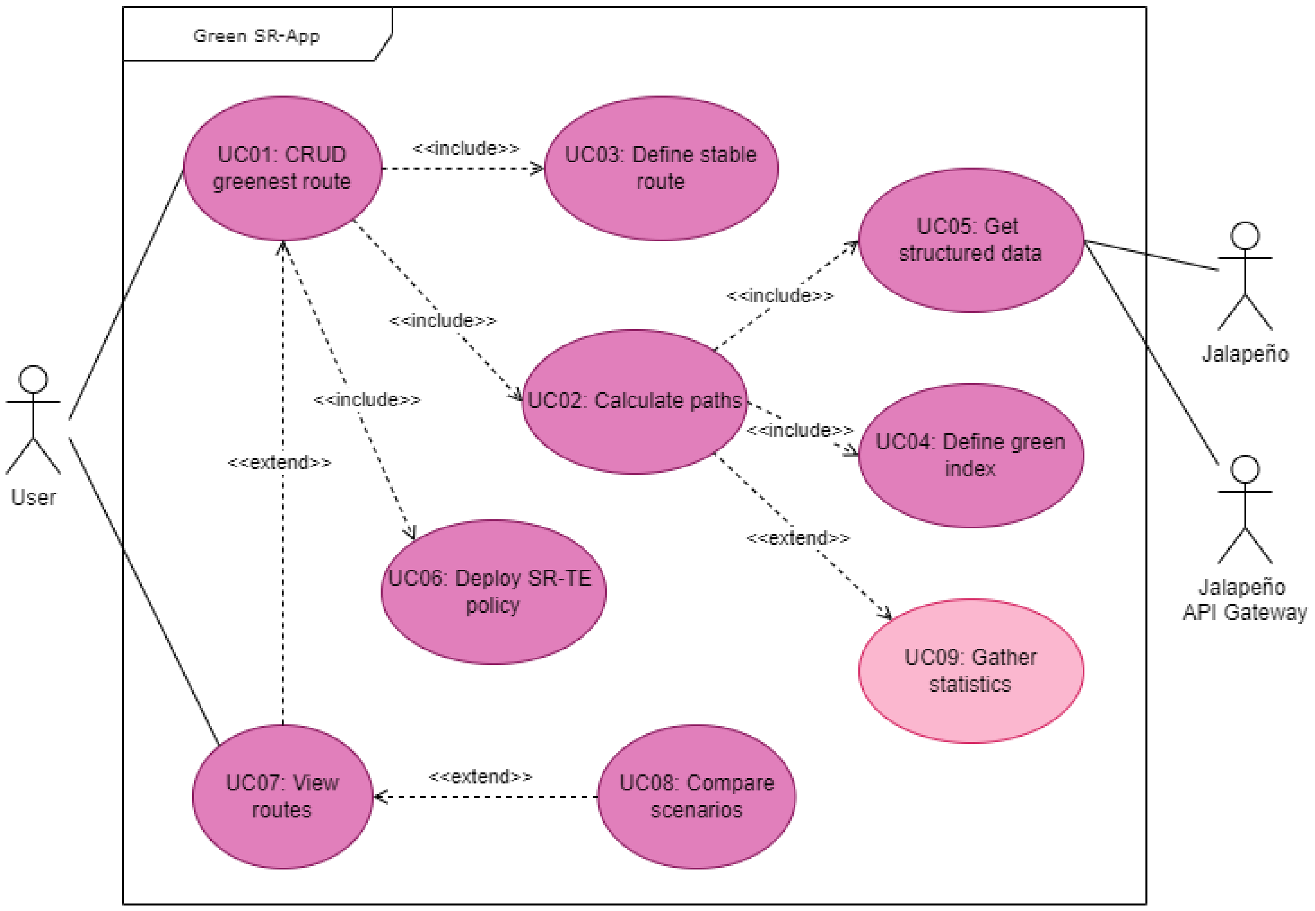


Figure 1.1: Use case diagram [1]

1.1.3 Use Case description

We will keep the format of our project thesis in which we specified a user story for each use case. The user stories follow the Connextra template, stated below.

"As a <role> I can <capability>, so that <receive benefit>"

Whereas the "so that" clause is optional and is only included if it adds value. By adhering to the template, all user stories follow the same pattern and provide a fast overview in an easy-to-read format.

We chose to use fully dressed use cases according to Larman for the more complex use cases. [2] This style was also taught at the Eastern Swiss University of Applied Sciences. We only used the fields that we considered to provide a benefit. We excluded the following fields from Larman's description: Scope, Level, Success Guarantee, Special Requirements, Technology and Data Variations List, and Miscellaneous.

The list numbering in the extensions sections always refers to the corresponding identifier of respective step in the Main Success Scenario.

UC01: CRUD greenest route

"As a User, I can get the greenest route over an API, so that I can lower my energy consumption."

Primary Actor	User
Overview	The User can access the Green SR-App over an API and request the greenest route by providing the ingress and egress node.
Stakeholders and Interests	The User wants an easy access over an API.
Preconditions	The Jalapeño API Gateway is delivering necessary source data.
Main Success Scenario	Create: 1. The User wants to create a new green route. 2. The User starts a new path calculation (detailed covered in UC02). 3. One or more green route(s) have been created. Read: 1. The User wants to access the available green routes. 2. The User performs a HTTP GET request with optional parameters. 3. The User receives a JSON list with one, multiple or all green routes. Update: 1. The User wants to update a current green route. 2. The User starts a new path calculation the same way as in the Create scenario. Delete: 1. The User wants to delete a green route. 2. The User performs a HTTP DELETE request with the id of the green route to be deleted. 3. The User receives a success response. For more details in general consult the API specification.
Extensions	Read: 1a. The User wants to access one specific green route. 2a. The User performs a HTTP GET request with a parameter which does not exist or could not be found.

	3a. The User receives a meaningful error message. Delete: 1a. The User wants to delete a green route which does not exists. 3a. The User receives a proper error message.
Frequency of Occurrence	As often as required.

Table 1.2: UC01 CRUD greenest route – Fully dressed use case

UC02: Calculate paths

"The Green SR-App can calculate the greenest (most energy efficient) route through a network."

Primary Actor	User
Overview	The path calculations take ingress and egress node as parameters and calculate the most energy efficient path between those two nodes.
Stakeholders and Interests	User wants to use the most energy efficient route.
Preconditions	The Jalapeño API Gateway is delivering necessary source data.
Main Success Scenario	1. The two parameters ingress node and egress node have been received. 2. Based on the k shortest paths algorithm, one or more greenest paths are calculated. 3. If one or more paths have been found, they will be returned.
Extensions	1a. If one or both parameters do not exist, an error is returned. 3a. If multiple equivalent paths have been found, all equal paths will be returned.
Frequency of Occurrence	The user-specified frequency or else when the green route regeneration interval time has been reached.

Table 1.3: UC02 Calculate paths – Fully dressed use case

UC3: Define stable route

"As a User, I expect that the calculated most efficient path stays stable for the duration I define."

Primary Actor	User
Overview	The User can specify at which frequency the paths are recalculated. The data is collected continuously but the paths will be calculated after the specified time has passed.

Stakeholders and Interests	The User relies on a certain stability of the routes. Additionally, if the paths change frequently, the overhead for calculation of manually triggering the recalculation will be too much.
Preconditions	The frequency must be specified. The data must be ready for consumption.
Main Success Scenario	1. After an event has been triggered the (re-)calculation of the greenest path is executed. 2. The result is logged into a storage service. 3. If the specified interval has passed, or if the topology has changed, the active path is updated.

Table 1.4: UC03 Define stable route – Fully dressed use case

UC04: Define green index

"As a User, I want to combine different metrics to a green index, which can be used to calculate the greenest path(s)."

Primary Actor	User
Overview	Before the greenest path can be calculated, it is necessary to choose the metrics, which should to be considered in the calculation. The so-called green index can now be composed by including those metrics.
Stakeholders and Interests	The User wants to have a green index based on at least three different metrics.
Preconditions	The data for the metrics is available.
Main Success Scenario	1. The metrics to be used are set at least by environment variables in the backend. 2. The selected metrics are considered in the next calculation of the green index, which in turn will result in a single number whereas a lower value is better.
Extensions	2a. One or more selected metrics have no data available and won't be considered in the calculation of the green index.

Table 1.5: UC04 Define green index – Fully dressed use case

UC05: Get structured data

"The algorithm needs the necessary fields from the Jalapeño API Gateway to calculate the most efficient route."

Primary Actor	Jalapeño API Gateway and Jalapeño

Overview	Get all the metrics over the Jalapeño API Gateway to calculate the best route.
Main Success Scenario	All the data regarding the topology and the available router sensors has been received by the Jalapeño API Gateway and the backend can calculate a green index.
Frequency of Occurrence	Topology: On demand or if the topology changes Telemetry data: On each new green route calculation

Table 1.6: UC05 Get structured data – Fully dressed use case

UC06: Deploy SR-TE Policy

"As a User, I can deploy the generated SR-TE Policy on the routers to steer traffic according to the defined rules."

Primary Actor	User
Overview	The User can configure and deploy the generated segment list(s) from UC02 to the ingress node to steer traffic along the greenest path.
Stakeholders and Interests	With the deployment and the previous use cases the network traffic saves energy by taking the greenest path.
Preconditions	UC02 was successful.
Main Success Scenario	1. The result of UC02 is a segment list of all the instructions required to follow the chosen path, which can be based on any metric. 2. The corresponding Yang Model is populated with the necessary data and is subsequently sent over gRPC to the ingress node. 3. The router receives the message and configures the policy to route the packets according to the chosen path.
Frequency of Occurrence	On every calculation of the path.

Table 1.7: UC07 Deploy SR-TE Policy – Fully dressed use case

UC07: View routes

"As a User, I want a simple web interface to see the chosen path(s) and be able to recalculate the routes."

Primary Actor	User

Overview	A web interface for the User to calculate the path(s) and view the ecological results. It displays the topology as well as the currently configured path.
Stakeholders and Interests	The User can present the possibilities of segment routing and the potential reduction of carbon emissions. For potential customers it's also possible to illustrate that it is possible to configure multiple routes for different groups of clients. As an ISP you can provide a green route for your customers who want to decrease their carbon footprint the same way as there are power plans based on renewables only.
Preconditions	All previous use cases are completed. The API is running, and the connection is established.
Main Success Scenario	1. The User connects to the frontend and sees the topology. 2. The User selects the available options based on the requirements and starts a calculation. 3. After the path has been configured, the now active path as well as a climate score and other metrics of the current path are displayed.

Table 1.8: UC07 View routes – Fully dressed use case

UC08: Compare scenarios

"As a User, I want to demonstrate the power savings of the greenest route(s) compared to the fastest route(s)."

Primary Actor	User
Overview	The interface must provide the possibility to showcase different scenarios by comparing e.g. the environmental impact of a green route with the path with least delay. It should be possible to calculate the green route as well as the fastest path in a network by clicking a button.
Stakeholders and Interests	By comparing different metrics the usefulness can be proven. If you compare the greenest route with the fastest route and you can show a significant difference in the climate impact, it is a viable product for ISPs to incorporate into their portfolio.
Preconditions	All previous use cases are completed. The API is running, and the connection is established.
Main Success Scenario	1. All steps from the main success scenario in UC07. 2. The User pins the result of step 1.

	3. The calculation with different metrics is conducted. 4. The results of steps 1 and 3 can be compared with each other.

Table 1.9: UC08 Compare scenarios – Fully dressed use case

UC09: Gather statistics

"As a User, I want historical statistics to understand the system and how it behaves."

Primary Actor	User
Overview	The User can generate reports based on the data the Green SR-App gathers and produces. This enables the User to observe the change of paths or how much power the system consumes at any point in time.
Stakeholders and Interests	The User can make further research and decisions based on the results of the reports.
Preconditions	To draw a graph with the variety, the path calculation must have been run a few times with realistic network data.
Main Success Scenario	1. Every time the greenest path is calculated, the result is stored in the database. 2. The telemetry data is collected continuously. The service reduces it to the relevant fields and enriches the data with some calculated information before it is written to the database or the caching service.
Frequency of Occurrence	Every time a green route is calculated.

Table 1.10: UC09 Gather statistics – Fully dressed use case

1.2 Product Backlog

Complementary to the use cases and NF requirements we use a product backlog which is ordered by priority. In each sprint planning we consider the items with the highest priority first and work down the list iteratively until we are either finished or run out of time.

Since the Green SR-App is primarily intended for demo purposes, we put our focus on new functionalities, rather than achieving the highest quality.

ID and priority	Description
1	Fully implement segments for SR into our SR-App.
1a	Prerequisite: Update to Jalapeño API Gateway version 1.3.
2	Switch from Jalapeño API Gateway requests (on demand) to JAGW subscriptions (live), through which our app receives notifications on topology changes.
2a	Only load the topology on changes instead of synchronizing them with every calculation.
3	Greenest path calculation algorithm version 2: a) Have the possibility to calculate multiple equal paths. b) Base the calculation on the watt per Mbit metric, instead of only the watt metric. c) Include at least three additional metrics into the path calculations.
4	Provide a graphical user interface to display the topology as well as the current path.
4a	Include the possibility to compare the environmental impact of different scenarios with different metrics and display the different routes taken.
5	Configure the network to steer the traffic along the calculated green route.
6	Deploy the application to the Kubernetes cluster from the INS.
7	Various optimizations of the current CI/CD pipeline.

Table 1.11: Product Backlog

1.3 Non-Functional Requirements

The non-functional requirements were defined according to the FURPS+ model. [3, pp. 56-57] The majority of the requirements were taken from the project thesis with a few adjustments and extensions.

1.3.1 Functionality

Security

Since the program will not be reachable from the internet, it is protected by the private network's existing security measures. There is no need to implement additional security measures.

Interoperability

The Green SR-App must interact with the Jalapeño API Gateway, which delivers the data required for the calculations. As long as the version upgrades do not imply any limitations, the most recent version will be used. If such limitations are discovered, the previously used version will continue to be used for future development and use. Additionally, the Green SR-App will configure the network to route the traffic along the green path, which will be tested on the newest version of the IOS-XR software with version 7.5.1. This enables us to use YANG models via gRPC to configure the router instead of using NETCONF or direct commands over SSH.

1.3.2 Usability

Understandability

The user should understand how the path was calculated. This will be achieved with statistics over the network and visual components in the web interface.

Operability

The system should behave in a predictable and secure way to prevent invalid configurations, which could lead to a breakdown of the network.

1.3.3 Reliability

Availability

Since this product mainly will be for demo purposes, no special measures will be taken to ensure a high availability. However, a big focus will be put on the fact that it should always run if it needs to run for demo purposes.

Recoverability

The application should be developed with the Cloud Native Standard in mind, which includes a simple redeployment, besides other guidelines, as described in chapter 5.2. [4]

1.3.4 Performance

Capacity

The application should be able to calculate paths in a network of up to 1000 nodes.

Time behavior

A new calculation of the greenest path should not take more than 10 seconds. This includes the configuration of the network as well as displaying the new route in the frontend. This is important for demo purposes, to prevent superfluous processing times while demonstrating the application to an audience.

1.3.5 Scalability

The backend should be scalable and agnostic to its hosting environment.

1.3.6 Maintainability

Analyzability

It must be possible to change the log level of the application. If the log level is set to debug mode, the application must write information about useful debugging events to the standard output.

2. Project Management

In this chapter we elaborate our project management methods, the involved stakeholders, how we planned our project time and which supporting tools we used. The milestones give a rough overview over our planned objectives. Later in the chapter assess the risks which could hinder our progress and define our mitigation methods and actions in the event of the hazard occurring.

2.1 Used Methods

The Rational Unified Process, shortly RUP, which is comprised of the four project phases: inception, elaboration, construction, and transition, is utilized as the basis project management method for our project.
The inception phase forms the start of the project and the definition of all project conditions. The first rough research and decisions are made to work out the initial project plan.
During the elaboration phase additional research is done to define the basic concepts as well as the software architecture and design.
The effective implementation takes place during the construction phase. The project begins to grow into a real product size and all the necessary features are implemented.
Finally, during the last phase, the transition phase, the product is introduced and delivered. This also includes the biggest part of writing this documentation.

We will also utilize Scrum with its iterations, especially due to the fact that during the construction phase iterative deliverable results are to be achieved. At Eastern University of Applied Sciences, the combination of RUP and Scrum is known as Scrum+, and it is intended to combine the benefits of the agile methods provided by Scrum with the classic approach of individual phases and milestones to effectively plan and execute the whole project.

2.2 Organization

The next two chapters list all team members and key stakeholders, including their contact information.

2.2.1 Project internal

Name	Profession	Email
Pascal Schlumpf	Software Developer	pascal.schlumpf@ost.ch
Jonas Hauser	Software Developer	jonas.hauser@ost.ch

Table 2.1: Project internal organization

2.2.2 Stakeholders

Name	Position or function	Email
Prof. Laurent Metzger	Supervisor	laurent.metzger@ost.ch
Julian Klaiber	Co-Supervisor	julian.klaiber@ost.ch

Michel Bongard	Co-Supervisor (until project week four) Developer of the Jalapeño API Gateway	michel.bongard@ost.ch
Severin Dellsperger	INS staff	severin.dellsperger@ost.ch
Francois Clad	Cisco Systems contact	fclad@cisco.com

Table 2.2: Stakeholders

2.3 Scheduling

The bachelor thesis consists of twelve ECTS – an equivalent of 360 hours work time per person and thus 720 hours as total project time. By following the agile approach approximately the entire time is utilized.

Due to various vacations during the spring semester at OST, the semester lasts 16 weeks. For this project, two additional weeks are intended, resulting in a total time budget of 18 project weeks.
During the first 17 weeks 20 hours per person will be spent for the project and, in the 18th week, twice as much (40 hours), due to it being a block week intended to work on this thesis and to achieve the total project time.
It is to be mentioned here that the duration during the project was extended by one week to compensate for a two-week absence from a student due to illness.

2.3.1 Important dates

Date	**Definition**
21. February 2022	Start of bachelor thesis and project kick-off
20. May 2022	Project interim presentation
13. June 2022	Deliver the abstract via the online tool
17. June 2022	Bachelor thesis exhibition at the OST
24. June 2022	End of bachelor thesis and project delivery
24. June 2022	Final presentation of bachelor thesis

Table 2.3: Important dates

2.3.2 Iterations / Sprints

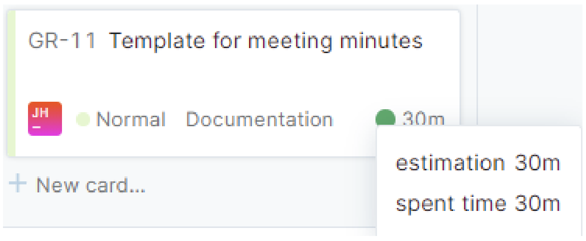
For this project, two-week sprints are used, since reasonably large work packages are created this way and positive experiences have been made with this in the term project. The exception to this is the block week, in which twice as much work is done. The last sprint therefore only lasts one week.

Each sprint generally ends on the respective Sunday and the following one starts on Monday, unless the team defines another exceptional timeslot due to absences or work limitations. All necessary Scrum meetings are defined in chapter 2.6.

2.3.3 Estimation and time spent

We estimate the workload for each work item and record it on the tickets in YouTrack during the sprint planning. The estimate is first made in relative units in the form of common clothing sizes (e.g. M) and then divided into absolute values by number of hours based on the time available for the sprint (e.g. 3 hours per task).

The time spent on the tickets is recorded according to the categories on the timetable in chapter 2.3.4, providing a clear picture of how many hours were spent on certain types of tasks.



estimation 30m
spent time 30m

Figure 2.1: YouTrack estimation and time spent [5]

A team member works ordinarily, between 16 and 24 hours per week, depending on the workload and other factors which influence the weekly required effort.

2.3.4 Timetable

Based on the project's characteristics, six categories of task types were defined. An approximate time estimate was made for each category to provide an overview, which can then be tracked on YouTrack using the time booked. In addition, a ten percent buffer was included in each sprint to absorb project volatility.
The defined milestones are explained in more detail in chapter 2.4.

	Inception		Elaboration		Construction													Transition
Study week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18*
Calendar week	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Project management	16		12		12		12		12		6		6	12		12		8
Documentation	40		12		8		8		8		4		4	8		8		48
Design / research	12		36		16		16		12		6		6	12		12		0
Development	4		8		24		24		28		14		14	28		28		4
Testing	0		0		8		8		8		4		4	8		8		4
CI / CD	0		4		4		4		4		2		2	4		4		4
Reserve (10%)	8		8		8		8		8		4		4	8		8		12
Total hours	80		80		80		80		80		40*		40	80		80		80*
Sprints	Sprint 1		Sprint 2		Sprint 3		Sprint 4		Sprint 5		Sprint 6		Sprint 6a	Sprint 7		Sprint 8		Sprint 9
Milestones		M1	M2	M3				M4					M5				M6	M7

*Project block week (40h per person)
**Only one team member working

Figure 2.2: Timetable [1]

2.3.4.1 Project management time composition

The following time boxes for project management related meetings have been defined and must be strictly followed under normal conditions.

Description	Effort (hours)	Total effort (hours)
Scrum ceremonies (meetings)	2.5	5
Meeting with supervisors	2	4
Additional meetings and administration	1.5	3
Total per sprint	**6**	**12**

Table 2.4: Project management time composition

2.4 Milestones

The seven milestones are defined in the following Table 2.5, along with the respective key objectives that go with them. The milestones were defined during the inception phase, meaning there may still occur changes based on the result of the requirements worked out up to M2 and the upcoming elaboration phase. The upcoming alterations will be indicated in a change history at the beginning of the project management section.

ID	Date	Title	Objectives
M1	06. March 2022	End of Inception	Definition of the initial project plan and the first specification of the development concept. All necessary tools for project management (YouTrack, Microsoft Teams and OneNote) are set up. Creation of a documentation template with the format and layout specifications as well as predefined chapters that will be covered.
M2	13. March 2022	Requirements	Finalization of requirements in the form of fully dressed use cases as well as functional and non-functional requirements. In addition a use case diagram is constructed which expands and builds up on the term project's use cases. Completion of the Product Backlog list with prioritized work packages.
M3	20. March 2022	End of Elaboration	Focus on research, decision making and first base concepts on the following topics: • Updating to the latest JAGW version • Implementation of segments • JAGW static requests to subscriptions migration • Definition of the green index v2

			• Conception for the most ecological path calculation algorithm v2 implementation • First PoC on how to present the results of our backend in a meaningful and easy to understand manner (intended for demonstration purposes) • Configuration of real network based on greenest path calculations And in addition, the ongoing continuation of this documentation based on the newly gained theoretical findings.
M4	17. April 2022	Features: Segments, JAGW subscriptions and algorithm v2	Based on the defined product backlog[1] with work packages sorted by priority (descending), the highest prioritized ones are picked and must be finished in a minimum beta version. This consists of the following: • Complete upgrading to the latest JAGW version and updating all other application dependencies • Full implementation of the capability to work with segments (segment routing) • Change the JAGW requests to subscriptions including a solution to live update and maintain our data from the connected network • Implementation of the greenest path calculation algorithm as a version 2 with new metrics involved • Optional: Optimization of our CI/CD pipeline • Optional: Deployment to our Server (Kubernetes)
M5	22. May 2022	Features: Digital presentation of produced results and configure network	The work items immediately following the highest priority ones mentioned in M4 are defined as objectives to reach a minimum beta state. This consists of the following: • Digital presentation of all produced results in a meaningful and easy to understand manner (intended for demonstration purposes)

[1] The product backlog is based on the requirements engineering in chapter 1

			• Possibility to configure the network to steer the traffic along the calculated green route • Capability to deploy the application to the Kubernetes cluster from the INS • Optional: Further steps for router telemetry data mocker
M6	19. June 2022	Green SR-App v1 release	The previously developed beta features must be worked out in detail so that they can be released as finalized version. Besides, several maintenance tasks and refactoring's are necessary to make the whole SR-App available as first full version v1.0.0.
M7	24. June 2022	Project delivery	The primary objective is to finish the whole project and deliver all necessary products. All documentations must be completed and in a final state.

Table 2.5: Project milestones

2.5 Responsibilities

Both students are equally empowered for this project from a general point of view and have contributed equally much to the project. Medium and large decisions are made by consulting with the other party. However, in order to increase efficiency and avoid work collisions or redundancies, everyone assumes different areas of responsibility as listed in the next table.

Team member	Responsibilities
Jonas Hauser	• Project management and meeting moderation • Lead for development concept • Web-API architecture and design • DevOps and infrastructure
Pascal Schlumpf	• Requirements management • Lead for research • Mocking and testing • Database design

Table 2.6: Team intern responsibilities

2.6 Meetings

Due to both students studying part-time besides working for other companies, the meetings will usually take place online via Microsoft Teams.

2.6.1 Scrum ceremonies

The following ceremonies will be held on Sunday evenings according to the timetable to close a Sprint (Sprint Review and Retrospective) and to start the next Sprint (Backlog Refinement and Sprint Planning).

Scrum Event	Sprint timing	Timeboxing
Sprint Review	End	0.75 hours
Sprint Retrospective	End	0.25 hours
Backlog Refinement	Start	0.25 hours
Sprint Planning	Start	1.25 hours

Table 2.7: Scrum ceremonies

2.6.2 Weekly exchange

Together with the whole project team, including the stakeholders defined in chapter 2.2.2, everyone is meeting weekly on Thursday at 10 a.m. to discuss the current work progress, clarify questions and discuss challenges. There is always at least one supervisor or co-supervisor present as well as one student. The industry partner is invited by the supervisor to join on demand to get insights and give feedback.
If necessary, additional meetings will be held with the INS staff on an individual basis.

2.7 Risk management

In this chapter, potential risks for the project are documented and classified. For each risk both preventive and reactive actions are assessed. After each sprint, the risks are reassessed and, if required, reclassified or removed if they have been eliminated completely.

The maximum damage potential is estimated based on the minimum resolution of half working days (four hours) in order to achieve realistic values in relation to ordinary working times. The probability of occurrence is classically estimated by percentages.

To reach the resulting weighted damage potential, the following calculation is performed for each risk and then rounded up to the next natural number:

$$Weighted\ damage\ potential = \ Damage\ potential * Probability\ of\ occurence$$

ID	Description	Damage potential	Probability of occurrence	Weighted damage potential
Ri1	A student absence due to illness or an accident.	40 hours	~~15 percent~~ 30 percent	**12 hours**
Ri2	Connecting to the necessary network components at the INS or Swisscom Lab is not possible.	60 hours	10 percent	**6 hours**
Ri3	The GitLab platform and or the CI/CD runners from INS are not or only partially available or overloaded. This risk is based	16 hours	50 percent	**8 hours**

	on experiences made in the term project.			
Ri4	The Jalapeño API Gateway from INS has unexpected bugs or limited use of functionalities.	20 hours	20 percent	**4 hours**
Ri5	It is not possible to configure a network with ASR 9000 routers as conceptualized in the elaboration phase.	40 hours	30 percent	**12 hours**
Ri6	We cannot implement the segment routing approach into our SR-App as conceptualized in the term project.	40 hours	10 percent	**4 hours**
Ri7	Throughput or other new metrics cannot be included as expected in the term project as metric for the second version of the greenest route calculation algorithm.	40 hours	~~30 percent~~ 10 percent	~~**12 hours**~~ **4 hours**
Ri8	The deployment to the Kubernetes environment from INS takes longer than expected.	8 hours	20 percent	**2 hours**
~~**Ri9**~~	Bad project planning and organization of the project.	~~40 hours~~	~~10 percent~~	~~**4 hours**~~
Ri10	One of Microsoft's collaborative work products (i.e. Word) has synchronization issues, resulting in data loss.	8 hours	5 percent	**1 hour**
Ri11	The new IOS XR image for the virtual lab will need a bigger time investment as expected to implement the router configurations.	20 hours	20 percent	**4 hours**
Ri12	One or more JAGW collections which are needed for the algorithm are missing.	8 hours	75 percent	**6 hours**

Table 2.8: Risk definitions

Adjustments after elaboration phase:

Ri7	Through extensive research during the elaboration phase, we gathered many possible metrics as well as ways to implement them. Therefore, it was possible to reduce the probability of the risk.
Ri12	After the discovery of missing collections in the JAGW we added the new risk.

2.7.1 Risk overview

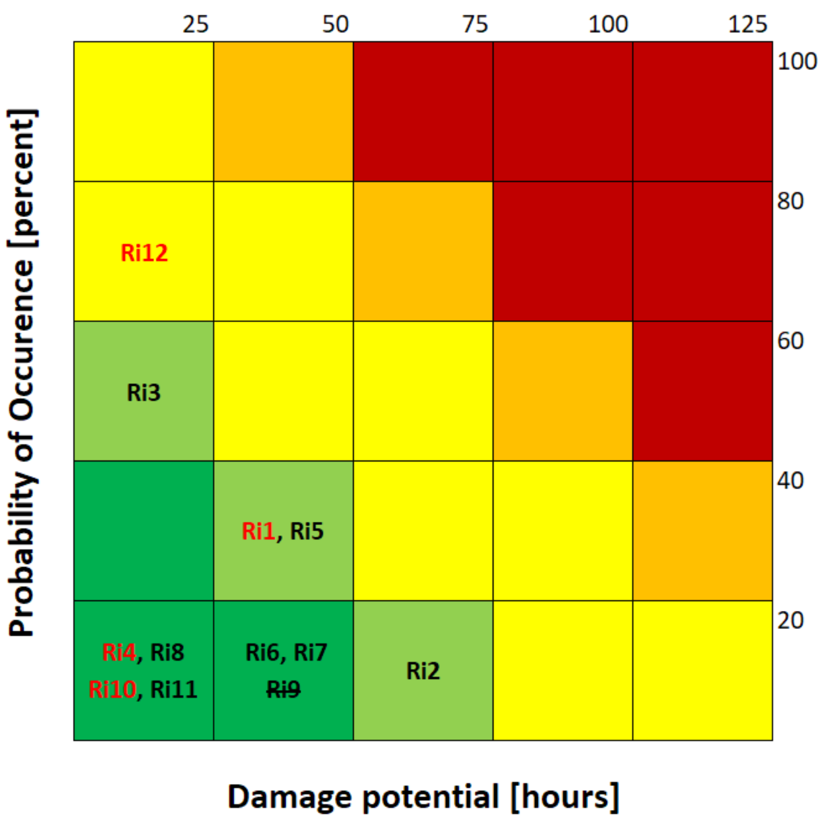


Figure 2.3: Risk overview [1]

2.7.2 Dealing with risks

As announced in the chapter introduction, for every risk some preventions and actions upon occurrence are carefully chosen and will reduce each risk significantly.

ID	Prevention	Behavior upon occurrence
Ri1	The students always follow the measures to prevent infection with the coronavirus based on the current regulations in Switzerland. The hygienic actions are implemented in the best viable way.	If a student is absent for a prolonged, interfering time, the next steps for the project will be discussed with the supervisors. Depending on the length of the absence due to illness, the project scope is shortened, or the time extended.
Ri2	The same tools as used in the term project are used to connect to the according network services. The tools have proven	We will always implement mocking layers for every dependency outside the application. This allows us to operate all the time, even in the event of an

	their usefulness. Another measure is to use direct and transparent communication channels to both INS and Swisscom so that system failures can be addressed quickly.	outage of required services (e.g. Swisscom Lab or the virtual network of INS).
Ri3	The mentioned platforms will be used as little as possible and replicated as close as possible locally (e.g. CI/CD build and check stages) to be less dependent. Git is a decentralized version control system, which means that the repositories are always completely mirrored at the individual nodes (including the personal devices). This allows complete temporary work without GitLab.	From experiences in the term project, overloads of the mentioned platforms often occur at specific times (approximately around 10 p.m.). If severe limitations occur in the same manner, work for which these services are needed is scheduled at off-peak times.
Ri4	The close and transparent contact with the developers from JAGW is a preventive measure in order to be able to solve occurring problems as quickly as possible.	The first step is to try to work around the bug without further ado or not to use the faulty part for the time being. In urgent cases, time from this project is invested to address fixes together with the INS.
Ri5	To minimize this risk early on, a minimal PoC is developed in the elaboration phase to test the most important and critical components in realistic terms. Furthermore, two networks are available to provide the necessary configuration and test possibilities. One is the virtual network from INS and the other is the Swisscom Lab. The components are built in such a way that highly functionality relevant parts can be mocked as fast as possible.	As already mentioned in the section on prevention, a brief amount of time is required to switch to the other network until the main test network is operational again. The second behavior is to finish and continue mocking all used components completely.
Ri6	The implemented domain model and entity relation concept will be reviewed again in the elaboration phase to confirm whether segments can be implemented on this basis. If discrepancies are found, the necessary modifications are conceptualized and planned at the earliest stage possible to avoid experiencing problems at a later point in time.	The necessary parts are rebuilt for the unexpected new demands via the simplest, yet appropriate way. If absolutely necessary, it is weighed whether the metric can be mocked in the SR-App or omitted entirely.

Ri7	In the upcoming elaboration phase, there will be a deliberate focus on the throughput metric in order to minimize the likelihood of mismatch as much as possible before the start of implementation. Furthermore, it is planned from the start to completely mock this metric to allow for isolated testing.	The extent to which this metric can be avoided, as well as the next most appropriate metric as an alternative, are evaluated.
Ri8	A know-how check is made with parties from INS as well as other students working on a thesis in the same environment. This way it can be ensured that potentially needed experience reports and other help can be found quickly.	Based on the prevention mentioned on the left, help is requested as promptly as possible to invest as less time as possible to deploy the app with all needed services.
Ri9	For this bachelor thesis, a very explicit structure for planning is used from the beginning, based on experience from the term project, previous projects, and guidelines. Work steps and planned activities are reviewed with the supervisors at regular intervals and an open feedback culture is encouraged. Solid project management experience from the students work environment is incorporated.	The project plan is adjusted and validated in detail to get the project back on track for success as quickly as possible. The organization is constantly improved as soon as individual difficulties arise, both internally and with the stakeholders.
Ri10	OneDrive and Word must be checked at regular intervals to ensure the automatic synchronization is still running. In addition, the documents and the notebook should be closed consistently when not in use, to reduce the chance of data loss.	In the case of data loss attempts are made to restore the necessary parts using Microsoft's restoring functionalities, depending on the volume of data lost. This must be done relatively quickly since the automatic temporary data backups have a short retention period.
Ri11	We will stay in close contact with the INS team to profit from their knowledge of the topic. Additionally, we will gather valuable information in a small PoC to mitigate the risks associated.	If all measures fail, we will go back to the earlier version of the image and continue working with that.
Ri12	We monitor the collections in JAGW to detect missing ones as soon as possible. This way they may be added in a later stage.	We will connect directly to the Jalapeño Arango database via the go connector for ArangoDB.

Table 2.9: Dealing with risks

2.7.3 Occurred risks

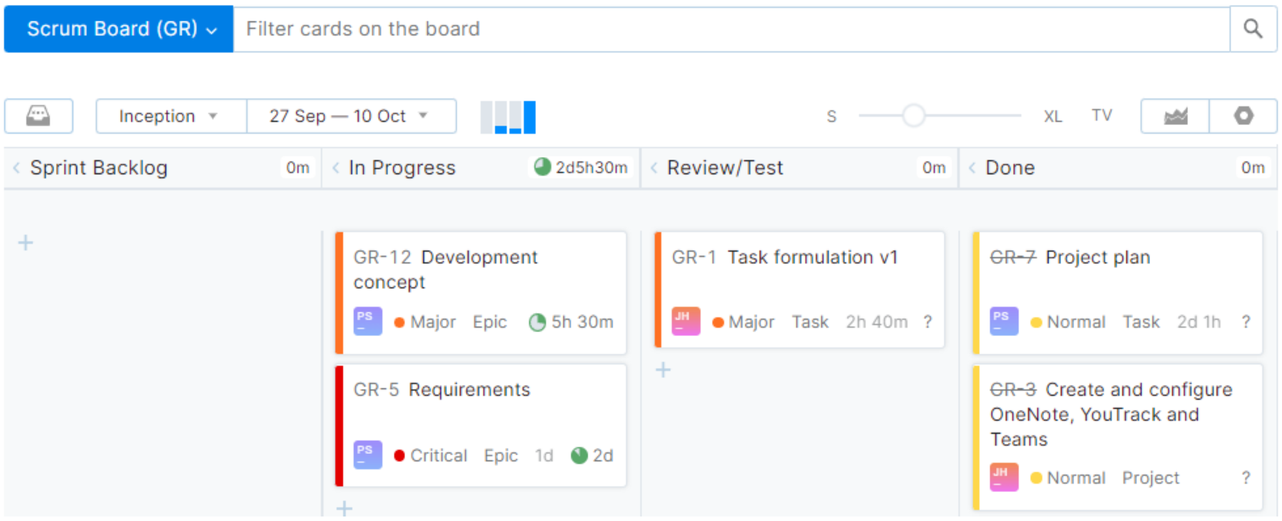
All occurred risks are beside the list below also highlighted with font color red in the risk overview in chapter 2.7.1.

ID	Extent of occurrence	Reaction and further steps	Real damage	Time of occurrence
Ri10	About 3 hours of documentation work was lost because parts in the document as well as the version history were overwritten during the device change between Jonas' devices. The exact cause is unfortunately not known but can be avoided by more consistent checks of the synchronization in the future.	The lost content in the documentation has been rewritten.	3 hours	02.03.2022
Ri12	The collection “l3vpn_v4_prefix” is not available in the JAGW.	We implemented our own database connection with our own queries to access the needed data from Jalapeño.	4 hours	17.03.2022
Ri4	The JAGW does not send any events if one subscribes to LsNodeEdges and any connection to the subscription service gets closed after 5 minutes. Additionally, there seems to be a problem with the new caching service, since the data returned by the request and the subscription service is inconsistent.	A lot of time had to be invested, to handle the missing events, to ensure that there is a connection active to receive events and other faulty behaviors.	16 hours (most likely more, but uncountable)	28.03.2022 and further on during the project
Ri1	Pascal took two weeks off during this work due to his acute illness. This resulted in a temporary time loss of 40 hours.	The bachelor thesis was extended by one week with 40 hours of work to compensate for the lost time.	0 hours (because the project was extended by one week)	02.05.2022

Table 2.10: Occurred risks

2.8 Tooling

We use the JetBrains product YouTrack for planning, coordination and tracking of work, though primarily only the functionality for Scrum Boards and time tracking. Positive experiences with this project management tool were made during the term project. Nevertheless, some trivial configuration improvements were implemented before the start of this project to brush away the last unpleasantness's. In the Scrum Board, estimations for each ticket can be recorded directly and the working time can be entered optimally according to the predefined categories, as visualized in the figure below.



Sprint Backlog 0m

In Progress 2d5h30m
- GR-12 Development concept — PS ● Major Epic 5h 30m
- GR-5 Requirements — PS ● Critical Epic 1d 2d

Review/Test 0m
- GR-1 Task formulation v1 — JH ● Major Task 2h 40m ?

Done 0m
- ~~GR-7~~ Project plan — PS ● Normal Task 2d 1h ?
- ~~GR-3~~ Create and configure OneNote, YouTrack and Teams — JH ● Normal Project ?

Figure 2.4: YouTrack Scrum Board [5]

Microsoft Teams is used for all communication, including its OneDrive and OneNote integration. OneDrive serves as data storage for project management and documentation belongings and enables a suitable environment for collaborative work in documents.
OneNote is used for various quick-living notes and as a knowledge base. Among other things, meeting minutes, raw data from research and a decision log can be found in this notebook.

To be able to work and especially develop efficiently the team members have their own powerful mobile and stationary work equipment with the necessary professional peripherals and tools installed.

2.9 Meeting minutes

We take brief minutes in German for each meeting with the topics defined in the following list. The date is not listed, since it is included next to the respective chapter title.

- Time
- Participants
- Agenda
- Notes on each agenda item discussed

As agreed by the supervisors, the minutes are generally not sent to the meeting attendees after it took place. If requested by a participant, access to a written protocol for a meeting held in the past could be granted. All minutes are listed in the appendix in part C.

3. Development

3.1 Version Management

We utilize GitLab, which is hosted by the OST, to exchange source code and application documentations, as well as the provided CI/CD solution. We have established a set of rules for working with Git in order to ensure a professional and efficient collaboration in the software development process within the team.

The branches are created and used according to the concepts of Git Flow (AVH Edition) [6]. In the two chapters below, we listed all the main and supporting branches used in the project and their respective purpose.
All branches are labelled in kebab-case, except for the release and hotfix branches which are written in semantic version numbers without a prefix. [7]

All commits must follow the Conventional Commits [8] guidelines. This ensures that the commit messages are uniform, describe the extensions and adjustments exactly, and that the change history is quickly traceable.

In addition, we use a code review strategy as described in detail in chapter 3.3.2.

3.1.1 Main branches

Branch name	Purpose
Develop	The primary branch in which the source code always reflects a state with the latest development changes for the next release.
Master	The productive branch in which the source code always reflects the current production state, tagged through the latest release version.

Table 3.1: Git main branches

3.1.2 Supporting branches

Branch name	Purpose	Base source	Merge destination
Feature	Used to develop new features for the upcoming or a distant future release.	Develop	Develop
Bugfix	The same as hotfix branches but it will be merged into the develop branch instead and is used to fix bugs made from feature branches.	Develop	Develop
Release	Support preparation of a new production release. It allows for minor prerelease fixes and preparing meta-data for a release (version number, build dates, etc.).	Develop	Master
Hotfix	Similar to release branches in that they are used to prepare for a new production release, however	Master	Master

	an unplanned one. They always use the master branch as the base.		

Table 3.2: Git supporting branches

We do not utilize support branches in this project.

3.2 Principles

Common software engineering best practices and concepts are always taken into account to ensure the maintainability and quality of the code base.

Acronym	Description
KISS	**K**eep **I**t **S**imple and **S**tupid
YAGNI	**Y**ou **A**ren't **G**onna **N**eed **I**t
DRY	**D**on't **R**epeat **Y**ourself
S.O.L.I.D	**S**ingle-responsibility principle **O**pen-closed principle **L**iskov substitution principle **I**nterface segregation principle **D**ependency inversion principle

Table 3.3: Software engineering principles

Furthermore, we generally and if reasonable follow the principles of the clean code principles.

3.3 Quality

3.3.1 Definition of Done

To achieve a better quality overall we integrated the use of DoD's with their easy-to-follow checklist to ensure that no important step has been forgotten.

3.3.1.1 Backend

- Coding Conventions (chapter 3.3.5) were respected
- No linter errors
- New potential integration and or unit tests introduced
- All integration and unit tests passed successfully
- Continuous Integration (chapter 3.3.6) went through without errors
- All findings from the code review (chapter 3.3.2) have been fixed
- The documentation was expanded or adapted if necessary

3.3.1.2 Frontend

- Coding Conventions (chapter 3.3.5) were respected
- No linter errors

- Continuous Integration (chapter 3.3.6) went through without errors
- All findings from the code review (chapter 3.3.2) have been fixed
- The documentation was expanded or adapted if necessary

3.3.2 Code Reviews

There are no modifications in the Git repositories that have not been approved during a review by at least one additional team member. The only exceptions are minor configuration changes and bug fixes, which must be implemented right away and don't introduce a potential risk. Via a pull request, the other team member is notified to review the respective change. We always work with supporting branches and only merge to a main development branch after a thorough code review.
A complete system test is be performed near the end of the construction phase as a second test strategy, the results are listed in the appendix of this document.

In addition to the two test strategies, manual tests are run repeatedly during the development process to ensure that the automated tests are working properly.

3.3.3 Testing

For our backend application and its potential surrounding systems, integration and unit tests are used as a first test strategy. Unit tests are a standard approach to test an application's functionality using various scenarios and can be conveniently integrated into the continuous integration process (chapter 3.3.6). Besides that, we use integration tests in some scenarios to reduce the count of unit tests and test the functionality of our SR-App from A to Z through all important components.

3.3.4 Static code analysis

A SonarQube instance was set up at the INS to be able to scan our static code. SonarQube can analyze code in a variety of programming languages and find errors, weaknesses, and bad practices, all of which may be viewed on a dashboard. This tool is also integrated into our continuous integration pipeline, allowing for automatic code analysis throughout the development process.

In the next chapter, we'll go over the code metrics we utilize and our reasoning behind them.

3.3.4.1 Code Metrics

We limit the SonarQube scanning procedure to the most relevant metrics that it employs. The primary metrics are detailed here, along with our goals for each.

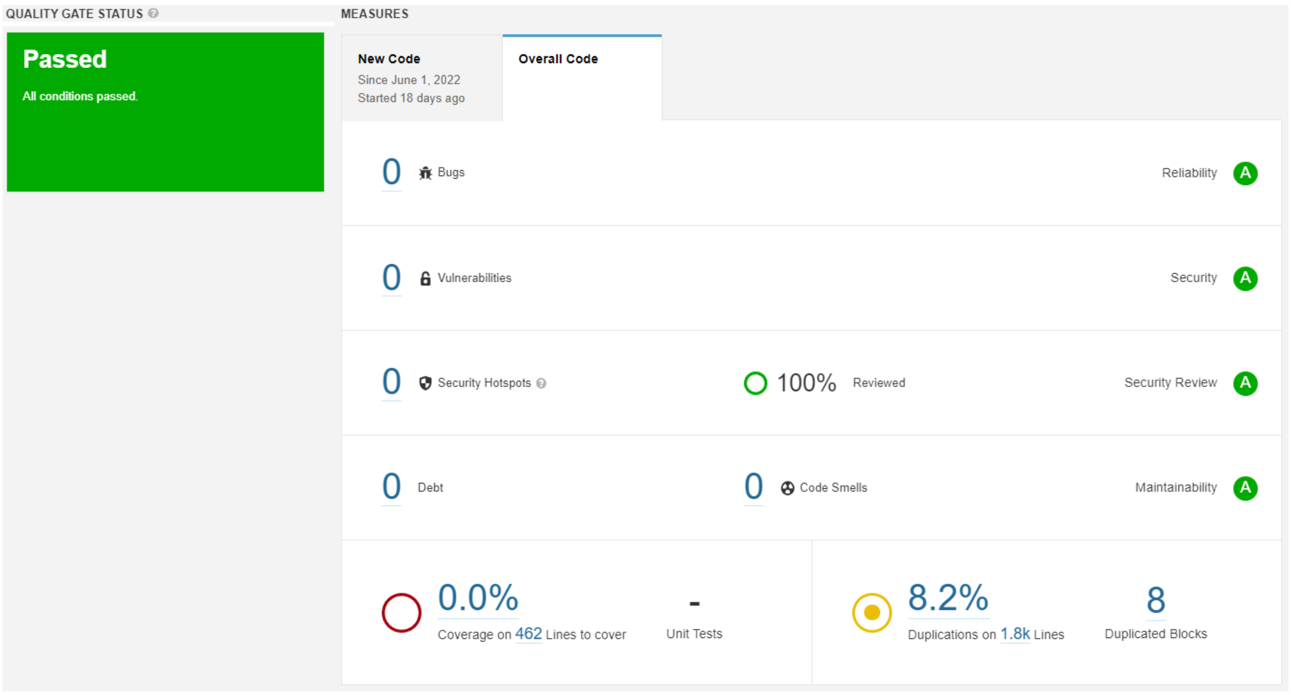
Name	Explanation	Goal
Bugs	Bugs are errors in the code that may cause the application to stop working or behave undesirably.	0
Vulnerabilities	An application vulnerability is a flaw or weakness in the system that can be exploited.	0
Security Hotspots	Code which needs manual checks to ensure that there are no security flaws.	0
Code Smells	Parts in the code which are hard to read and understand.	0

Test coverage	Defines the percentage of how much code is tested by integration and unit tests.	min. 80%
Duplications	Defines the percentage of how many lines of code are duplicated over the whole application.	max. 1%

Table 3.4: Quality code metrics

Frontend

The next screenshot shows the final quality gate status from SonarQube for the frontend. Due to the complexity of the network graph components which uses Sigma.js to view the graphs, certain components could not be reused, causing a duplication statistic of around 8.2%.


QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code
Since June 1, 2022
Started 18 days ago

Overall Code

0 Bugs — Reliability A

0 Vulnerabilities — Security A

0 Security Hotspots — 100% Reviewed — Security Review A

0 Debt — 0 Code Smells — Maintainability A

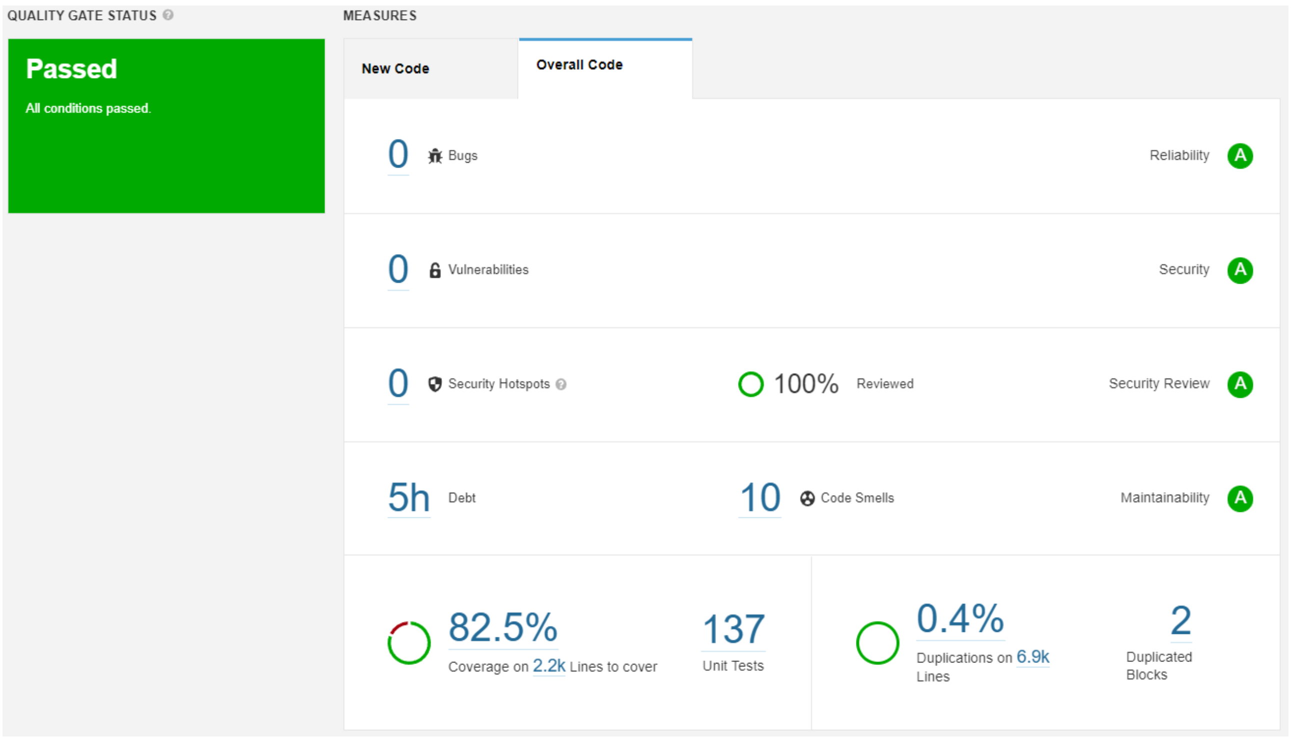
0.0% Coverage on 462 Lines to cover — - Unit Tests

8.2% Duplications on 1.8k Lines — 8 Duplicated Blocks

Figure 3.1: SonarQube frontend quality gate status [1]

Backend

The next screenshot shows the final quality gate status from SonarQube for the backend. We've reached a total coverage of 82.5%, which is an improvement of around 15% in comparison to the term project. Unfortunately, there are ten code smells remaining because SonarQube does not yet fully support our used Go version 1.18. All files that use generics cannot be parsed properly and were falsely detected as Code Smell.


QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code | Overall Code

0 Bugs — Reliability A

0 Vulnerabilities — Security A

0 Security Hotspots — 100% Reviewed — Security Review A

5h Debt — 10 Code Smells — Maintainability A

82.5% Coverage on 2.2k Lines to cover — 137 Unit Tests

0.4% Duplications on 6.9k Lines — 2 Duplicated Blocks

Figure 3.2: SonarQube backend quality gate status [1]

3.3.5 Coding conventions

3.3.5.1 Frontend

For the frontend, we did not include any special coding conventions. We instead follow the common best practices for TypeScript and the React framework.

To format files, Prettier is used, which is widely used code formatting tool in the JavaScript ecosystem. [9]

In addition, the compliance with structuring rules is ensured by the linter ESLint. ESLint is configured for the use of TypeScript and React along with the standard checking rules. [10]

3.3.5.2 Backend

We follow the coding conventions of the CockroachDB, which define a wide range of distinct requirements for styling, performance, and best practices for Golang, which is the primary programming language used in our project. [11]

These best practices are complemented by two tools. The first one is golangci-lint [12] which applies a large set of rules through different single linters every time a file is saved. These rules need to be followed strictly and are enforced at Git commit time. Some default rules were deactivated because they hindered the progress while not providing enough benefits.

As an additional quality increasing measure, we also make use of gofmt which formats an entire source file according to standard formatting rules every time it is saved. [13] Thanks to better readability it is easier to spot errors and prevent future bugs.

3.3.6 Continuous Integration and Deployment

For our continuous integration and deployment, we utilize the GitLab integrated CI/CD feature with the runners provided by INS. Those runners provide us with faster run times, since we do not need to share the runners with the other users of the OST.

There are three Docker image strategies in the pipeline. A build-only image containing the source code and configurations is used for linting, testing, and SonarQube scanner. The other two strategies are for development and production deployment builds only, do not contain source code (just the executable), and are reduced to a reasonably small size.

In the following table all stages are listed and described chronologically. The frontend and backend pipelines are comprised of the same stages, except that the frontend doesn't have a test stage as it does not include any automated tests.

Stage	Description	Fail level	Execution Target
build	Builds the backend based on the build only image[2] and tags it with the latest commit hash.	Build failure	Every commit
lint	Executes the entire linting process for the build image and displays the results.	Linting errors (not warnings or info's)	Develop, release, hotfix, and merge requests
test	Executes all integration and unit tests.	If at least one test fails	Develop, release, hotfix, and merge requests
sonar-scanner	Executes the SonarQube check via sonar-scanner.	If a code metric does not pass the quality gate (described in chapter 3.3.4.1)	Develop, release, hotfix, and merge requests
registry-cleanup	Removes the previously created build only image[3] from the container registry to keep it minimal in size.	No directly associated fail level	Develop, release, hotfix, and merge requests
build-dev	Builds and tags the development deployment build image[4] with "develop".	No directly associated fail level	Develop
build-prod	Builds and tags the production deployment build image[5] with "latest"	No directly associated fail level	Master (implicit only tags)

[2] Dockerfile name: «Dockerfile»
[3] Dockerfile name: «Dockerfile»
[4] Dockerfile name: «Dockerfile.deploy»
[5] Dockerfile name: «Dockerfile.deploy»

	and the respective version of the application (semantic version).

Table 3.5: CI/CD stages

The deployment is performed manually with a predefined Helm chart using the required image and other services. The integration of the deployment into the pipeline itself would have been beyond the scope of this work due to various challenges of the infrastructure used and was therefore not implemented.

3.4 Error Handling

3.4.1 Input validation and output sanitization

To avoid potential errors within the application, all data objects for creation or updating via API endpoints are validated using properly specified data fields and data types on predefined data transfer objects (DTOs). The API only interacts with DTOs for ingress and egress data; therefore, app internal entities will never leave the interface. This prevents the exposure of internal data over the API that is not meant to be accessible via the API.

3.4.2 HTTP status

Whenever possible, the backend replies with an appropriate HTTP status code and/or error message. In unusual behaviors, extraordinary errors respond with the code 500 for Internal Server Error. Only fatal errors cause the program to stop working altogether. All other failures that are not specifically handled, as well as so-called Go panics, are handled automatically by the Gin Web Framework without crashing the application.

In the table below you will find a basic example about the previously described criteria.

Request	Response Header	Response Body

GET /node/8	HTTP **404 Not Found**
	
	Content-Type: application/json
	Content-Length: 28
		{
		"**error**": "element(s) not found"
		"**identifier**": "8"
		}

Table 3.6: HTTP response sample

3.4.3 Logging

To satisfy the eleventh point of the Twelve-Factor Methodology and to have a solid logging capability, an extended logger was implemented instead utilizing Go's default logger and registered as the main logger on all other included frameworks and libraries. This logger offers seven log levels, ranging from most serious to least serious: fatal, panic, dpanic, error, warn, info, debug. Only serious problems will force the program to terminate, as mentioned in the previous section 3.4.2. The log service writes its output directly to the operating system's standard output. The target log level can be adjusted by environment variables at program start.

3.5 Development Tools

The following table lists the main tools used for development and briefly describes their purpose.

Tool name	Purpose
JetBrains GoLand	Integrated development environment which is specialized for Golang development. Extended by several plugins from the JetBrains marketplace.
JetBrains WebStorm	Integrated development environment which is specialized for web development with HTML, CSS and JavaScript/TypeScript. Extended by several plugins from the JetBrains marketplace.
Insomnia	Modern open-source API client to test endpoints.
Docker	Provides a standardized and stateless toolchain to easily set up environments in form of containers. Extended by docker-compose for interconnecting tools and services.
Git Bash or WSL	Bash emulator or full virtualized Linux environment to execute and test binaries and have a production like operating system.
GitLab	A remote version control system with extra functionalities like GitLab CI/CD, merge requests, integrations with Kubernetes, integrations with YouTrack and other DevOps tools.

Table 3.7: Development tools

4. Domain analysis

4.1 Domain Model diagram

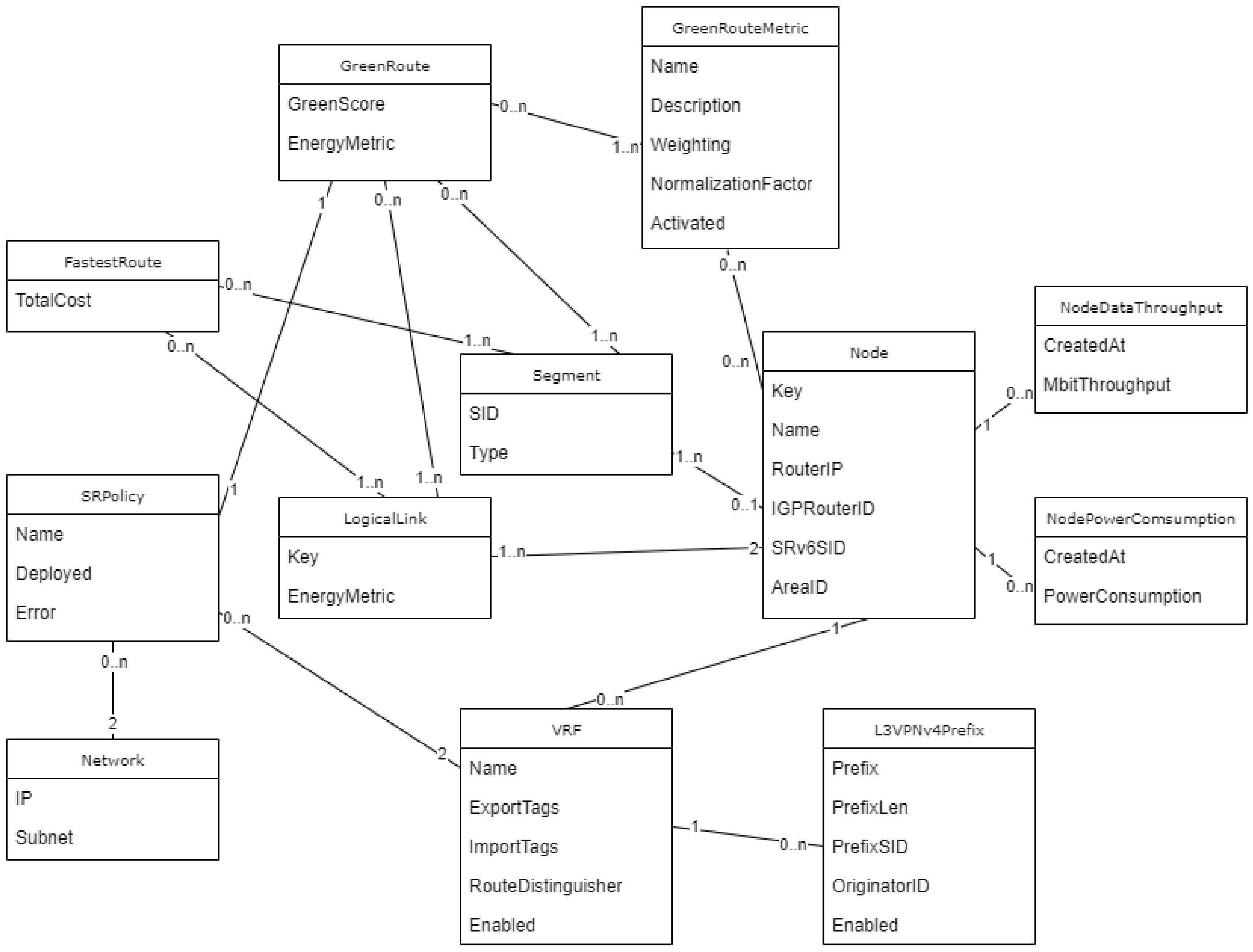


Figure 4.1: Domain Model diagram [1]

4.2 Domain Model explanation

4.2.1 Node

A Node is an ASR 9000 router in the segment routing domain which actively participates in segment routing. For this thesis we limit the scope to the ASR 9000 router from Cisco with the IOS-XR software. Each node has green metrics which are either directly read from the device over sensors or collected manually. This is solved over the GreenRouteMetrics, which handles all metrics as well as the NodePowerConsumption and NodeDataThroughput, which represent the data which has been read from the device sensors.

Each Node has multiple ingress and egress links to other nodes which are contained in the LogicalLinks. To create the segment list used in segment routing each node has one Segment entity.

4.2.2 VRF

VRF or Virtual Routing and Forwarding is a technology where multiple routing tables co-exist to enable separated routes for each customer group. This works by importing export tags, which each router exports, as import tags by another router used in the VRF. Each separate route has its own route distinguisher which enables the exported routes to be recognized.

4.2.3 L3VPNv4Prefix

L3 VPN is a VPN which lives on the OSI Layer 3. It utilizes IPv6 and VRFs to create and manage user data. It usually is used in backend VPN infrastructures, e.g. for connections between datacenters or back offices.

The L3VPNv4Prefix table on the ArangoDB in Jalapeño contains information on the connected customer networks for each node as well as the used VRF. This information is required in the configuration of the calculated green routes on the router.

4.2.4 NodePowerConsumption

The NodePowerConsumption contains the measured power consumption of the nodes in watt. Each Node has a history of all measured values gathered since the application started to fulfill the purpose of collection statistical data.

4.2.5 NodeDataThroughput

The NodeDataThroughput contains the measured throughput of the nodes in Mbit/s. Each Node has a history of all measured values gathered since the application started to fulfill the purpose of collection statistical data.

4.2.6 Segment

The Segment is composed of the IPv6 address of the Node as well the Node's type. It is used in the segment list required for the segment routing configuration.

4.2.7 LogicalLink

Each LogicalLink has a source ("from") node and a destination ("to") node and is therefore directed. For the calculation of the green route as well as the fastest route the combined value of all the metrics from the destination node will be saved in the energy metric field. This value represents the edge weight for the Yen K-shortest paths algorithm. Klicken oder tippen Sie hier, um Text einzugeben.

4.2.8 GreenRoute

On the GreenRoute the ingress node and the egress node for each calculation will be stored. The resulting list of LogicalLinks and segment list can also be found on the green route. This is the basis for the SR-Policy which is required for the configuration. Each GreenRoute contains also the sum of all green scores to compare the different GreenRoutes.

4.2.9 GreenRouteMetric

For each node there exist one or more GreenRouteMetrics. These have a value, a weighing factor and a normalization factor to adjust the values to the same order of magnitude. Each metric can be turned on or off. All metrics of a node combined result in the green score for the specific router.

4.2.10 SRPolicy

The SRPolicy combines all the calculated and acquired data for an ingress node and egress node pair. It is the central entity to configure the route on the router. If multiple green routes have been calculated, each one gets its own SRPolicy. Additionally, a separate policy is created for each VRF. Thus, if two equal routes have been calculated and there are two VRFs, then four SRPolicies will be generated.

4.2.11 Network

The Network entity provides the IP address for the configuration via SSH. It contains the IPv4 address and the respective subnet. The IP address is required to build up the connection to the router.

5. Architecture and design specification

5.1 System overview

To give the readers a visual overview over the existing software and the further developed app, several diagrams according to the C4 model were created.

Due to the API Gateway notifying the Green SR-App on topology changes and new telemetry data, the Green SR-App does not need a special cache to handle the current data. It is possible to request the necessary data and process it on demand. This way it is possible for multiple instances to have the same results for the calculation since all instances request and receive the same data.

The following C4 system context diagram shows the user, as well as the developed Green SR-App and the Jalapeño API Gateway which accesses Jalapeño from Cisco Systems.

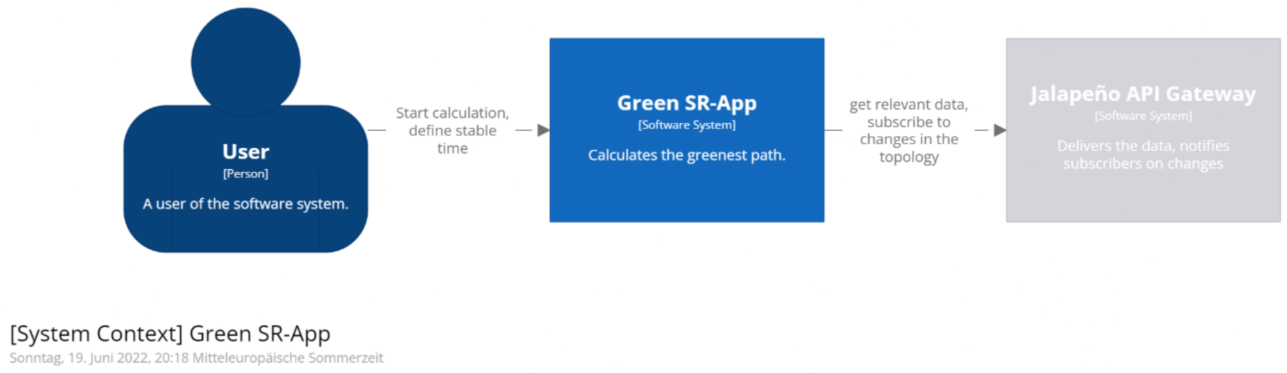

User
[Person]
A user of the software system.
Start calculation, define stable time
Green SR-App
[Software System]
Calculates the greenest path.
get relevant data, subscribe to changes in the topology
Jalapeño API Gateway
[Software System]
Delivers the data, notifies subscribers on changes
[System Context] Green SR-App
Sonntag, 19. Juni 2022, 20:18 Mitteleuropäische Sommerzeit

Figure 5.1: C4 model system overview [1]

The following C4 container diagram shows how the different containers interact with each other and which protocols are used. In addition, the diagram shows how the user communicates with the software system. The external systems Jalapeño API Gateway and Jalapeño are not visualized in detail, since we only use their provided services.

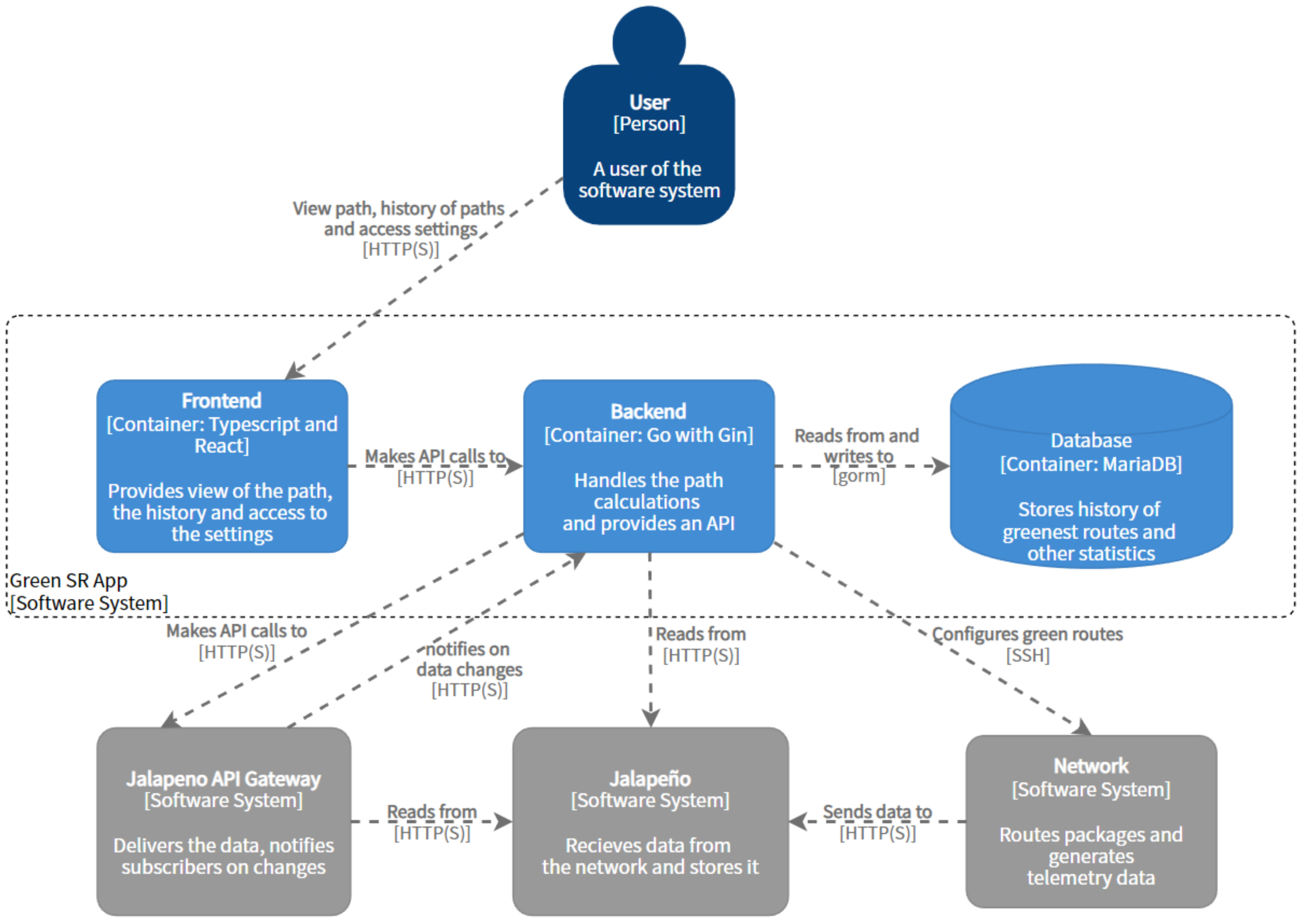

User
[Person]
A user of the software system
View path, history of paths and access settings
[HTTP(S)]
Frontend
[Container: Typescript and React]
Provides view of the path, the history and access to the settings
Makes API calls to
[HTTP(S)]
Backend
[Container: Go with Gin]
Handles the path calculations and provides an API
Reads from and writes to
[gorm]
Database
[Container: MariaDB]
Stores history of greenest routes and other statistics
Green SR App
[Software System]
Makes API calls to
[HTTP(S)]
notifies on data changes
[HTTP(S)]
Reads from
[HTTP(S)]
Configures green routes
[SSH]
Jalapeno API Gateway
[Software System]
Delivers the data, notifies subscribers on changes
Reads from
[HTTP(S)]
Jalapeño
[Software System]
Recieves data from the network and stores it
Sends data to
[HTTP(S)]
Network
[Software System]
Routes packages and generates telemetry data

Figure 5.2: C4 model container system overview [1]

5.1.1 Jalapeño API Gateway

The Jalapeño API Gateway enables the backend to access the stored date of Jalapeño in an easy and standardized way over a gRPC API. Additionally, the Gateway API notifies the backend on changes in the topology or on new telemetry data. This way the application can react to changes and can calculate a new path if necessary.

5.1.2 Jalapeño

Jalapeño is a software from Cisco which collects and processes telemetry data and topology data of the connected network and stores them in an InfluxDB for telemetry data and in an ArangoDB for topology data.

5.1.3 Network

The network is the real connected computer network, which has multiple routers and routes packages, and generates telemetry data.

5.1.4 Frontend

A user who uses the Green SR-App will access the app via the frontend, in which it is possible to view the calculated route representing the most energy efficient way. For comparison, it is also possible to calculate and display the fastest route. The user can trigger a new calculation if necessary. In the settings there is an option through which the user can define which metrics should be used to calculate the green route.

5.1.5 Backend

The backend is the heart piece of the whole application. Here, the data from the API Gateway is evaluated and prepared for path calculation. When the data is in the required form, the path calculation is executed, and the most efficient path(s) are evaluated. The result will be consumed by the frontend and stored in the database for later statistical analysis.
If the user wants to apply the route to the network, it is possible to deploy it. The path calculation gets triggered when the defined retention duration has passed. To keep energy costs low, the calculation is only executed when needed. To get a fair measurement of the efficiency of the router not only the power consumption is considered but also the throughput. This way newer and more efficient routers perform better in the benchmark even if they consume more electricity.

5.1.6 Database

In the database the calculated routes will be stored for future statistical analysis. In addition, the application stores other potentially interesting metrics to create reports on. The reports will not be part of this thesis. It was decided to use a MariaDB because it offers the possibility to store graph data as well as relational data. It works well with Go and it is possible to run it in memory if we need the performance boost. This way we can store the chosen route as a graph and the power data in a relational table.

5.2 Twelve-factor app methodology

The frontend and backend applications are designed following the 12-factor methodology in order to be entirely cloud native standard compliant. [4] In the next subchapters, this methodology is used to analyze the complete applications, including its environments and services, step by step.

5.2.1 Codebase

“One codebase tracked in revision control, many deploys” [4]

The version control system used is GitLab, which also enables for direct integration of CI/CD and connects to YouTrack. The backend and the frontend are developed in separate repositories with completely separate code and building process to a fully distinct development and production environment purpose. No direct staging environment is used because the development environment performs similarly.

The backend and the frontend could each be verified completely independently using the 12-factor methodologies.

Evaluation: fulfilled

5.2.2 Dependencies

“Explicitly declare and isolate dependencies” [4]

With version 1.12, Go added a new dependency management system that documents each dependency and its peer dependencies in a file called go.mod. This file also contains definitions for which version to use, as well as peer dependencies. With version 1.12, they added the ability to create modules, which we utilize in the backend to isolate the code clearly and logically.

In the frontend, the NPM ecosystem is typically used to manage dependencies. However, not the NPM tooling itself is used, but the alternative Yarn.

Evaluation: fulfilled

5.2.3 Config

"Store config in the environment" [4]

All static configurations for the frontend and backend are defined via environment variables. They're configured locally by the developer using preset environment files, and then injected in the Helm chart deployment using system environment variables.

In the Golang context, the library GoDotEnv is utilized to fully achieve this. Every dynamic variable that varies between environments is either set locally via an .env file or injected via the Helm chart definition for the needed environment. On launch, the application loads the environment variables, which may afterwards be used by the application.

In the JavaScript context, all environment variables are provided by default via the JavaScript inbuilt process.env context. Just as in the backend, an .env file can be used for the required variables or via the Helm chart for other environments.

Evaluation: fulfilled

5.2.4 Backing services

"Treat backing services as attached resources" [4]

MariaDB is accessed by a URL that is made up of different values from the environment file. As long as the underlying database is still a MySQL database provider, it is possible to replace it. If necessary, the entire repository layer can be replaced to use a different type of database. There are no code modifications required if you want to utilize a different MySQL provider.

Evaluation: fulfilled

5.2.5 Build, release, run

"Strictly separate build and run stages" [4]

The backend CI/CD is divided into seven distinct stages. They are covered in detail in the 3.3.6. In summary, there is one build step for checking, three quality stages consisting of linting, unit testing, and static code analysis and two build stages for development and production. The deployment is done manually based on needs for each environment separately. The pipeline clearly distinguishes build, release, and run, and traversing the path the other way is either impossible or unforeseeable when following the right workflow.

The three key stages of GitLab, GitLab CI, and Kubernetes deployment result in a well-defined path. The first is for version control, while the second is for continuous integration and deployment. The operating portion is then performed on the server using Kubernetes.

Evaluation: fulfilled

5.2.6 Processes

"Execute the app as one or more stateless processes" [4]

The app must operate in a stateless mode. This means that it must not preserve any states during runtime, except when saving for a brief amount of time, such as when data is being processed further. Data saved for a limited period of time, on the other hand, should never be meant for a future request. All data that has to be kept for an extended period of time must be kept in linked services such as databases.

The program is fully stateless since we always perform calculations based on the telemetry data that Jalapeño currently provides. This may be one or more telemetry entries based on the time range provided or the network uptime up to this point. The relational and synchronized database stores all data intended for long-term storage. Short-term data is only kept on a per-request basis.

Evaluation: fulfilled

5.2.7 Port binding

"Export services via port binding" [4]

Each service provided must be tied to a port by rule. It must not rely on webserver runtime injection. The web app must provide HTTP as a service by binding a port and listening on that port for ingress traffic.

This is accomplished in Go by direct port binding, which is also fully supported by the Gin Web Framework. For HTTP traffic, the app just requires one port. The same is true for the frontend with React.

Evaluation: fulfilled

5.2.8 Concurrency

"Scale out via the process model" [4]

One or more processes represent each running program. Web applications can use a variety of distinct types of process execution.

The computationally intensive operations, which includes the synchronization of Jalapeño and the following computation of the green route with required steps in between, was accomplished by parallelization using semaphores and mutexes. Go includes built-in parallelization features that are quite elaborate. All other sections of the program, even with excessively big volumes of data, are already fast enough without parallelization. Go also supports vertical scaling of processors, which is compatible with the Green SR-App.

Evaluation: fulfilled

5.2.9 Disposability

"Maximize robustness with fast startup and graceful shutdown" [4]

App processes should be discardable. This implies they must be able to start and stop in a brief period of time. Furthermore, the program must be able to be turned off gracefully.

The processes were designed to be streamlined and lightweight, allowing for rapid startup and shutdown. When a shutdown request is received, it is not guaranteed that all continuing queries and computations have been finished. In addition, it is not possible to deprovision the configurations made on the network and restore them to their previous configuration. To achieve this, time resources were lacking in this thesis.

Evaluation: partly fulfilled

5.2.10 Dev/prod parity

"Keep development, staging, and production as similar as possible" [4]

This rule demands that the environments be as comparable as possible. The environments local, development and production are utilized in this project.

The following table compares traditionally produced applications to twelve factor apps, with an extra assessment for the Green SR-App developed in this thesis.

	Traditional app	**Twelve-factor app**	**Green SR-App**
Time between deploys	Weeks	Hours	Hours to days
Code authors versus code deployers	Different people	Same people	Same people
Development versus production environment	Divergent	As similar as possible	As similar as possible

Table 5.1: Dev/prod parity comparison

The dependent services, such as databases, are retained and the differences in building and creation are reduced to a minimum.

Evaluation: fulfilled

5.2.11 Logs

"Treat logs as event streams" [4]

A twelve-factor app should never be in charge of sending or storing produced logs. Every running process must transmit its logs straight to the standard output as event streams, with no intermediate steps.

Without any intermediate steps, the backend publishes all logging information straight to the output stream (including standard output and standard error). The log level can be changed using environment variables.
The frontend interacts in the browser and publishes possible errors via the browser console.

Evaluation: fulfilled

5.2.12 Admin processes

"Run admin/management tasks as one-off processes" [4]

It is about single tasks that must be executed on the productive system, such as starting database migrations. These activities must always be carried out on as similar systems as feasible.

Such tasks are reduced by utilizing containers for all application components. The container can be simply redeployed in the case of changes or adaptations. Migration scripts are occasionally required for particular services; however, they are always tested first in the development system, which is as close to the production environment as possible.

Evaluation: fulfilled

5.3 Technologies

In the next two chapters, the most important technologies used in the frontend and backend are listed and categorized. It should be noted that smaller, less important libraries/extensions have been omitted deliberately, as the listing of these is not relevant.

5.3.1 Frontend

Component	Technologies and Frameworks	Libraries/Extensions
Frontend	• TypeScript • React • Material UI	• React-sigma v2 (underlying Sigma.js) • Graphology • Axios • Classnames • Sass • Faker
Frontend development support		• EsLint • Prettier • Commitlint

Table 5.2: Used frontend technologies

5.3.2 Backend

Component	Technologies and Frameworks	Libraries/Extensions
Backend	• Golang • Gin Web Framework • Gonum Numerical Packages • Jalapeño API Gateway	• Gin-contrib/zap • Gin-contrib/cors • Jinzhu/Copier • Joho/GoDotEnv • Stretchr/Testify • Swaggo • Zap • gRPC and protobuf • Zapgorm2
Internal database	• MariaDB	• Gorm • Gorm MySQL driver
External database	• ArangoDB	• AragoDB Go driver

Backend development support	• Golang • Docker and Docker Compose	• Go run, build, fmt, test, clean, install, get, mod • Joho/GoDotEnv • Golangci-lint • Swaggo

Table 5.3: Used backend technologies

5.3.3 Programming Language

In the preceding term project, it was decided to use Go as our programming language for the backend. We were quite satisfied with the performance and native concurrency of Go. Therefore, and the fact that a change would cost too much of time, we keep the programming language.

On the frontend side, there is less freedom to choose the programming language since there is one big player. To follow the most modern standards, TypeScript was used rather than JavaScript, which is also widely used in combination with the React framework.

5.3.4 Web Framework

Gin Web Framework is a Go framework that uses a Martini-like API but is up to 40 times faster than Martini. Gin is appropriate for APIs when performance is critical. Furthermore, this web framework provides middleware and a crash-free API. As a result, it can detect and recover from panics in the Go context. As a result, the server is as much as possible accessible. Gin, in addition to the previously described features, can quickly verify JSON and improve routes.

5.3.5 Storage

MariaDB was already used as a persistent storage in the term project. With the continuation of the project through this bachelor thesis, the change to a real graph database was considered and extensively analyzed in the elaboration phase.
The decision was made to not change anything regarding the persistent storage, since the risks for an unreasonably high additional effort outweigh the advantageous aspects. MariaDB can also still meet all requirements for this thesis in a sufficiently performant manner.

The use of Gorm has proven itself and no major or unsolvable problems have ever been encountered with this ORM. Due to the positive experiences through the term project, Gorm will continue to be used.

5.3.6 Development support

5.3.6.1 Linter's runner

There are several linters for Go, each of which covers a specific portion or serves a certain purpose. The golangci-lint Go linters runner is used in this project so that they do not all have to be integrated and executed separately. This linters runner allows you to run all available linters and customize them separately if necessary. [12]
All Go cli inbuilt linters and useful recommended linters are utilized at least. The complete list of linters and checkers used can be found in the appendix.

5.3.6.2 Testing

Since the unit tests are not run by the linter's runner, they must be run individually. The Go command "go test" is used to accomplish this. Additionally, a library called Testify is used to make the definition of unit tests easier. Testify mostly benefits from wounding assertion checks, which are commonly used in unit tests. [14]

5.3.6.3 OpenAPI specification

To document the API, Swagger is used, which follows the OpenAPI standard. Since version 3 of this standard is not yet fully supported by Go, version 2 was used. A CLI tool allows to automatically generate the API documentation using annotations as comments in the code.

5.3.6.4 Container virtualization

To provide the code as an image and to replicate the pipeline functionality locally, docker is used with in conjunction with with docker-compose.

5.4 Backend Architecture

This chapter provides an overview of the backend architecture and illustrates the links between the various components. We distinguish between the frontend, which we do not go into depth about, the backend, which entails the business logic and APIs, and the database.

With the three layers of application, domain model, and infrastructure, we follow the domain driven development paradigm. Our controllers, which make up the API, are located in the application layer. We handle the models in the domain in various entity files, as well as the data transfer objects (DTO). Finally, there is an infrastructure layer, which contains the services with the business logic and the repository.

We also put the logger and the environment (config) in the infrastructure layer. We have a Go specific cmd container that serves as the application's entry point.

The common container handles migrations and seeds that are not utilized during regular runtime but rather to manage database changes and offer a set of default data in the database.
The frontend received the necessary data via the API. The API obtains the required data from the services using data transfer objects.
Any configuration data is delivered through environment files, and all containers that utilize configuration data must consequently contact the environment container. The same is true for the logger. The logger is accessible to all containers that need to log information.

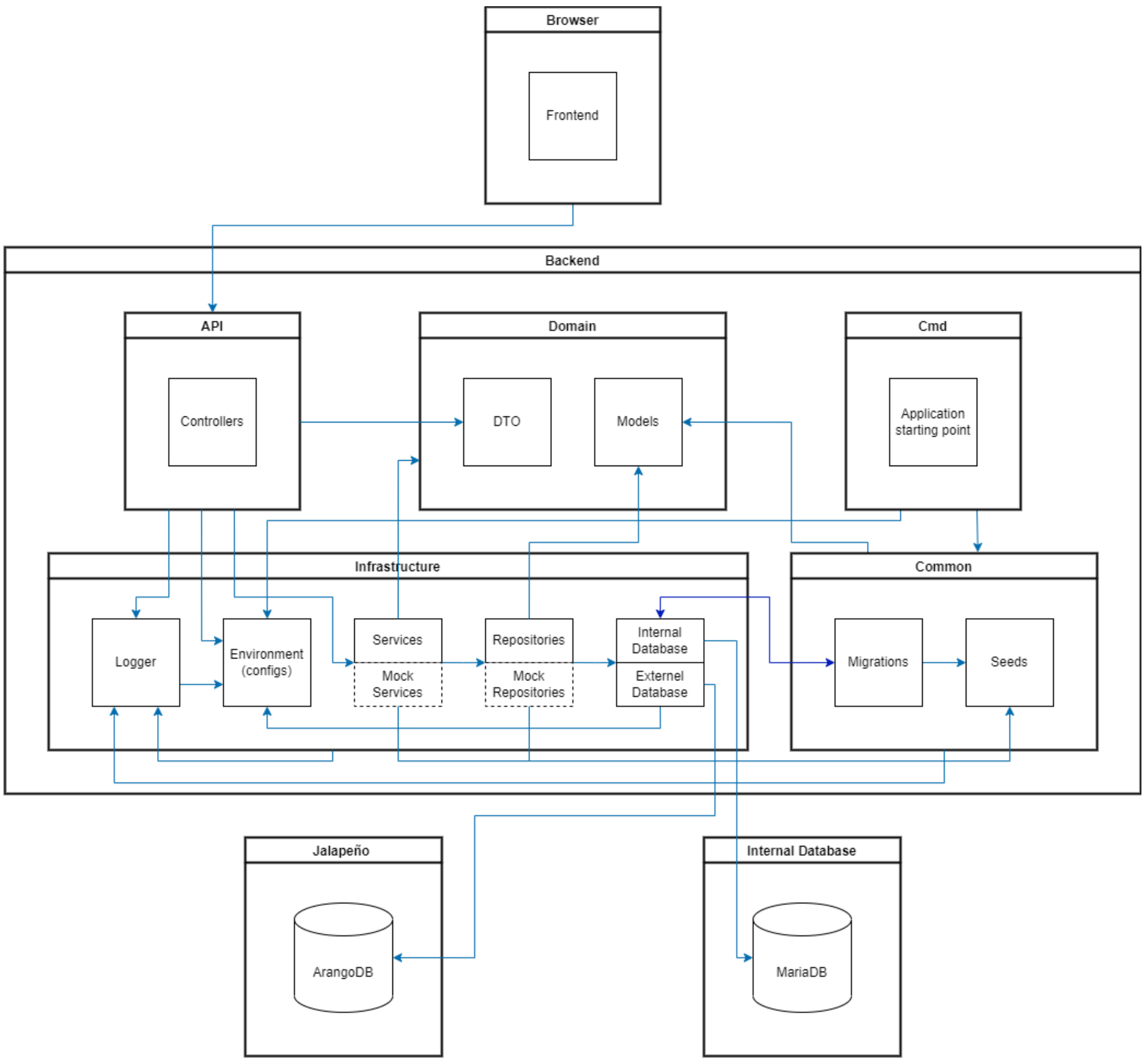


Figure 5.3: Backend architecture [1]

5.4.1 Backend

The API, domain, infrastructure, and common sub-modules form the Green SR-App backend. Each of these sub-modules contains at least one, if not numerous, packages. The sub-modules and their packages are explained in the following chapters.

5.4.1.1 API

The sub-module API consists of a single package in which all application controllers are defined as a single file per HTTP endpoint.

Controllers are the initial point of contact for an incoming request. They also specify what endpoints look like and what functions and operations they provide. They are also in charge of managing the input and output, as well as validating the input. HTTP error codes are also defined here based on service results and other characteristics.

5.4.1.2 Domain

The package DTO includes all data transfer objects that are utilized as transfer data objects by controllers and services. These DTOs are basically objects that exist between the layers and transmit their created data between them.

Models is a package containing multiple entities. Entities describe what data for persistent storage should look like. They also indirectly specify the database schema's appearance.

5.4.1.3 Infrastructure

The infrastructure sub-module contains the majority of the packages discussed thus far. There are five packages in the setup, most of which are exclusively for the internal application part.

The most essential package, services, includes all of the business logic for each service. The services receive controller requests and return the calculated result. Communication with repositories is almost always required in order to access persistent data. They oftentimes do not pass on the data directly, but rather include the relevant business logic, including any necessary computations. Individually utilized mock services implement the same logic as regular services, but with realistic mock data.

The repositories package is probably the second most essential package in this sub-module after the services package. The repositories handle direct database queries. They are provided by the ORM library by default and provide the most important CRUD queries to the database without any extra definition. When non-standard specified queries are required, new repository definition files containing raw queries or query languages are developed. Individually utilized mock repositories implement the same logic as regular repositories, but with realistic mock data.

The third package, database, initiates and configures the database connection. It provides the repositories with an active database connection.

The fourth package in the bundle is named environment, and it is responsible for loading and delivering environment variables defined in an environment file or directly from the system's environment variables.

Finally, the package logger serves as the main logging service of the application. It handles the whole logging of the program into standard output and standard error. It provides several log levels for various purposes and environments. An environment variable can be used to specify the log level.

5.4.1.4 Cmd

This sub-module contains no directly existing packages. In Go projects it is typical to store the application's start in a package named cmd. Since this package contains no further program logic, it is sometimes referred to as the "main"-function in other programming languages.

5.4.1.5 Common

All packages that do not belong in the Domain Driven Design context and are not utilized in the usual runtime are placed in this container. The migration manages database schema changes, whereas the seed populates the database with initial or test data in a test or development environment. It is likely that further packages will be added in the future.

5.4.2 Internal Database

The internal database, which is a relational MariaDB, is housed in this container. It saves all computed pathways as well as other statistical information. It is accessible by Gorm, which is a Go ORM library found in the repositories package.

5.4.3 Jalapeño

Jalapeño is addressed directly as an external database, because the Jalapeño API gateway does not offer all required data sets. Therefore, in special cases, a direct data relationship must be applied.

5.5 REST API

The API was designed in accordance with API development standards and best practices and does not diverge from them. Because both developers are familiar with these standards and best practices, no other external sources were consulted to complete the API specification. Making the API endpoints as practical and easy as possible was a primary focus during development.

The specification of the POST requests for calculating the greenest and fastest paths, generating SR-Policies and deploying them deserves special attention. The developers purposefully chose to indicate the resource to be generated in the request path. The alternative of passing information via the body was dismissed since the data to be created is entirely computed by the backend rather than by the user making the request.

5.5.1 API responses

All of the API's potential responses are explained in detail in the table after the rough list of DTO's below. In the default success scenario, the requested resources are returned as data transfer objects, although in some instances, an empty answer is acceptable.
Individual DTO items are offered as follows:

- BaseDTO
- FastestLogicalLinkDTO
- FastestRouteDTO
- GreenLogicalLinkDTO
- GreenRouteDTO
- GreenMetricTypeDTO
- GreenMetricTypeUpdateDTO
- L3VPNv4PrefixDTO
- LogicalLinkDTO
- NetworkDTO
- NodeDTO
- NodePowerConsumptionDTO
- RouteDTO
- SegmentDTO
- SrPolicyDTO

For more detailed information the source code of the Green SR-App backend should be consolidated.

Definition	Status type	HTTP status code	Response object with fields
Default	Success	200	Individual DTO or empty response
Base	Success	200	BaseResponse • Message[6] (string) • OpenAPIDocs path (string)
Telemetry data	Success	200	TelemetryDataResponse • FakedData (bool) • TelemetryData (PowerTelemetry or ThroughputTelemetry object)
Not found	Failure	404	NotFoundResponse • Error[7] (string) • Identifier (string)
Internal Server Error	Failure	500	ErrorResponse • Error (string)
Not Implemented	Failure	501	ErrorResponse • Error[8] (string)

Table 5.4: API responses

[6] Response text: «Hello from the Green-SR App API»
[7] Response text: «element(s) not found»
[8] Response text: «Not supported due to FAKE_ALL_DATA activated»

5.6 Entity relationship diagram

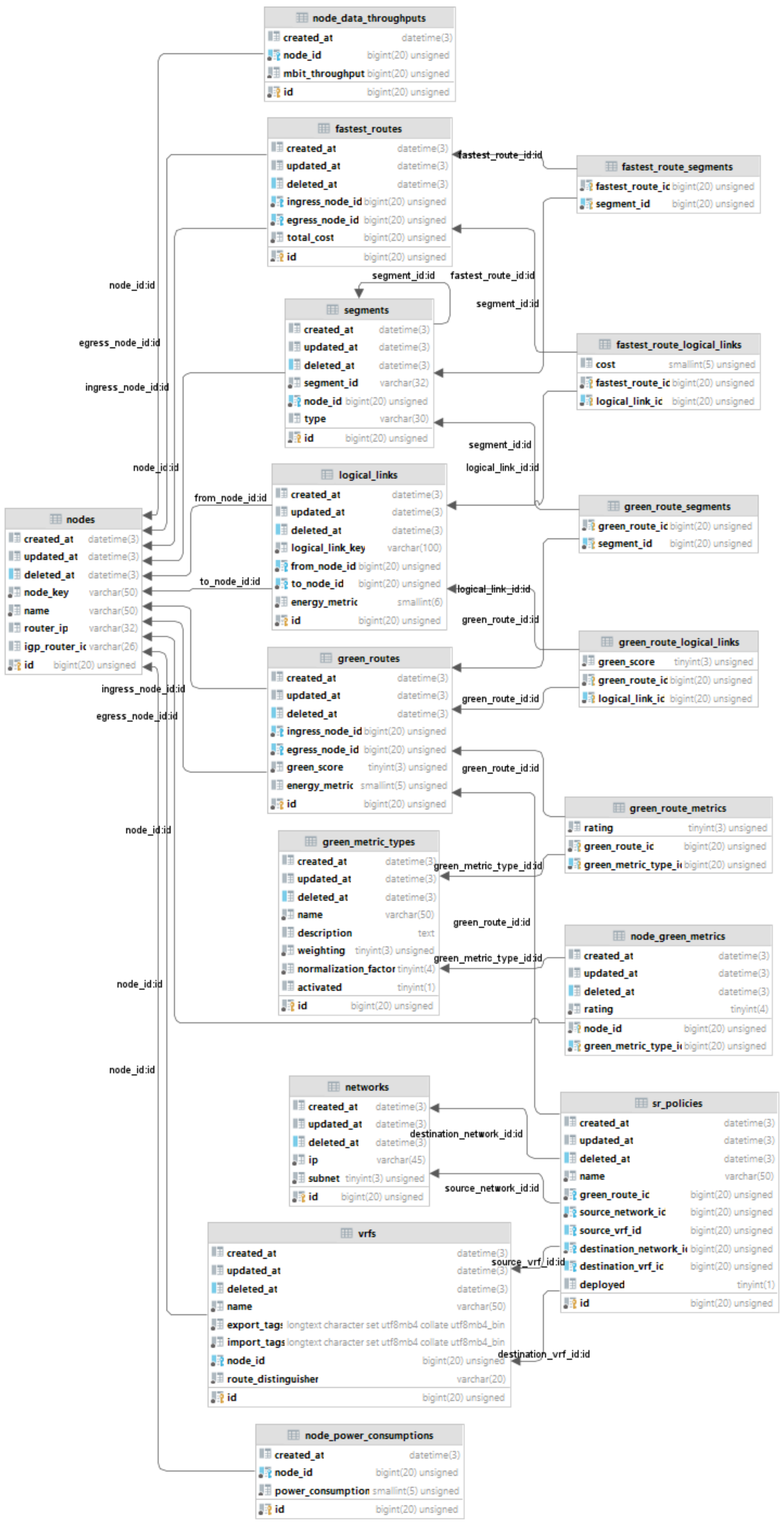


Figure 5.4: Entity relationship diagram [1]

5.7 Infrastructure

5.7.1 INS virtual Lab

Kubernetes was used to create up a server on the INS network, which provides the necessary services for this project via multiple Kubernetes pods. The following environment has been set up, as specified in the technical report in part A:

- Webserver (Nginx)
- Jalapeño API Gateway
    - Request and subscription service
    - Cache and cache service (Redis)
    - Proxy (Envoy proxy)
- Jalapeño
    - Time Series database (InfluxDB)
    - Graph database (ArangoDB)
    - Telemetry collector and processor (Telegraf)
    - Topology collector and processor (Gobmp)
    - Event streaming (Kafka)
    - Observability platform (Grafana)
    - Configuration management (ZooKeeper)

The latest versions of the new Green SR-App are also always deployed to this server.

The INS put up a virtual lab to obtain the routers (Cisco ASR 9000) and the necessary data required for this study. Using SSH over port 22 provides access to all devices. In the next table, you'll find the virtual devices that form the lab.

Name	Type	Intended use
XR-1	Router	Base ASR 9000 like router that routes traffic on the network with segment routing.
XR-2	Router	Base ASR 9000 like router that routes traffic on the network with segment routing.
XR-3	Router	Base ASR 9000 like router that routes traffic on the network with segment routing.
XR-4	Router reflector	Special ASR 9000 like router reflector which listens on all other router updates. Main data source for Jalapeño.
XR-5	Router reflector	Special ASR 9000 like router reflector which listens on all other router updates. Main data source for Jalapeño.
XR-6	Router	Base ASR 9000 like router that routes traffic on the network with segment routing.
XR-7	Router	Base ASR 9000 like router that routes traffic on the network with segment routing.

XR-8	Router	Base ASR 9000 like router that routes traffic on the network with segment routing.
Cust-A-ZRH	Customer Network entry	Sample private customer network on one side of the whole network.
Cust-B-ZRH	Customer Network entry	Sample private customer network on one side of the whole network.
Cust-A-BSL	Customer Network entry	Sample private customer network on one side of the whole network.
Cust-B-BSL	Customer Network entry	Sample private customer network on one side of the whole network.
Cust-A-ZRH-PC1	Customer PC (Ubuntu)	Computer within the customer's private network. Its main purpose is to generate traffic on the network.
Cust-B-ZRH-PC1	Customer PC (Ubuntu)	Computer within the customer's private network. Its main purpose is to generate traffic on the network.
Cust-A-BSL-PC1	Customer PC (Ubuntu)	Computer within the customer's private network. Its main purpose is to generate traffic on the network.
Cust-B-BSL-PC1	Customer PC (Ubuntu)	Computer within the customer's private network. Its main purpose is to generate traffic on the network.

Table 5.5: INS virtual Lab equipment

Declaration of Independence

I hereby declare,

- that I have carried out the present work myself and without outside help, except for that which is explicitly mentioned in the assignment or agreed upon in writing with the supervisor.
- that I have mentioned all sources used and cited them correctly according to common scientific rules of citation.
- that I have not used any material protected by copyright in this work in an unauthorized way.

Pascal Schlumpf

Place and Date:

Rapperswil, 23.06.22

Signature:

[illegible]

Jonas Hauser

Place and Date:

Rorschach, 23.06.22

Signature:

[illegible]

Rights of use

Agreement

Subject

This agreement regulates the rights over the use and further development of the results of the bachelor thesis Green Routing by Jonas Hauser and Pascal Schlumpf under the supervision of Prof. Laurent Metzger.

Copyrights

The student is entitled to the copyrights.

Usage

The results of the work may be used and further developed by both of the students, the OST and Cisco Systems after completion of the work.

Student	**Student**	**Supervisor**
Pascal Schlumpf	**Jonas Hauser**	**Prof. Laurent Metzger**
Place and Date:	Place and Date:	Place and Date:
Rapperswil, 23.06.22	Rorschach, 23.06.22	Rapperswil, 23.06.2022
Signature:	Signature:	Signature:
[illegible]	[illegible]	[illegible]

List of Figures

List of Tables

Glossary

ArangoDB	ArangoDB is a free and native open-source database system with multiple models. It supports three data models (key/value, documents and graphs) with a database core and a unified query language called AQL (ArangoDB Query Language).
Container	Processes running on the host system or hypervisor, but in a strictly delimited context. Often used in the context of Docker or Kubernetes.
Data transfer object	A data transfer object (DTO) is an object that carries data between processes and application layers.
Docker	Docker is free software for isolating applications using container virtualization.
Domain driven design	Domain driven design is an approach to modelling complex software. The modelling of the software is significantly influenced by the technicalities to be implemented in the application domain.
ECTS	European Credit Transfer System to accumulate study achievements.
ESLint	ESLint statically analyzes JavaScript and TypeScript code to quickly find problems.
Gin	Gin Web Framework is an API framework for Go.
GoBMP	Is basically an implementation of Open BMP (RFC 7854) protocol's collector in Golang.
Golang	A statically typed, compiled programming language designed at Google.
Gorm	An Object Relation Mapper for Go.
Graph Database	A graph database is a database that uses graphs to represent and store heavily interconnected information.
Green index	The green index contains all green scores and ranks all routers by their ecological aspects. It is the measurement data basis for the green route calculation.
Green score	A green score is a numeric value between 1 and 100, where 1 is better and 100 worse in sense of how ecological something is. It is merged by multiple metrics which measure ecological aspects.
gRPC	gRPC is a modern open-source high performance Remote Procedure Call (RPC) framework that can run in different environments. It can efficiently connect services for multiple purposes.
Helm	Helm charts help define, install, and upgrade Kubernetes applications and is basically a package manager for Kubernetes.

InfluxDB	InfluxDB is an open-source database management system, specifically for time series concepted. It is developed and distributed by the company InfluxData.
Insomnia	Simple and open-source API Client. Insomnia is an alternative for the well-known tool Postman, which is also an API client.
Jalapeño	System developed by Cisco, which collects and processes data from attached networks, including telemetry data.
JetBrains YouTrack	Issue tracking and project management system from the manufacturer JetBrains. Alternative for other brands like Atlassian Jira.
Kafka	Kafka, developed by Apache, is an open-source distributed event streaming platform used for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.
Kaniko	Kaniko is a tool to build container images from a Dockerfile.
Kubernetes	Is a professional open-source system for automating the deployment, scaling and management of container applications.
Martini API	Martini creates standards-compliant APIs with native OpenAPI 3.0 support to improve API discoverability and management. Basically, it is a set of tools to build and consume web APIs for this standard.
Merge Request	Also known as Pull Request. Workflow in a version control system to make source code changes and review them.
Metric	A standard of measurement.
Network Topology	Describes the arrangement of systems on a computer network.
Object-relational mapping	Object-relational mapping (ORM) is a programming technique for converting data between incompatible type systems using object-oriented programming languages.
OpenAPI specification	Defines a standard interface to RESTful APIs for both humans and computers
Panic	In Golang, panic is just like an exception in other known languages. It arises at runtime. In other words, panic means an unexpected condition occurred in a Go program due to which the execution of the program is terminated.
Protocol	A protocol is a standard set of rules that allow electronic devices to communicate with each other.
Rational Unified Process	Procedure model for software development projects divided in four main phases.
Redis	Is an open source, in-memory data structure store, used as a database, cache and message broker.

Scrum Board	Is one of the tools used when applying the Scrum project method. Basically, a board filled with work items.
SonarQube	SonarQube is a platform for static analysis and evaluation of source code quality. It analyzes the source code regarding various quality areas and presents the results rated in quality gates via a dashboard.
Sigma.js	Sigma.js is a modern JavaScript framework that allows you to see and interact with network graphs in your browser. It interacts with graphology, a multifunctional graph manipulation package, in a symbiotic manner.
Standard Error Stream	Via standard error a program can output error data via error stream. This is often used to display error logs in a command line interface.
Standard Output Stream	Via standard output a program can output data via data stream. This is often used to display logs in a command line interface.
Statis Code Analysis	Static code analysis (SAST) is a static software testing procedure performed at translation time of software. The source code is subjected to a series of formal checks that can detect certain types of errors.
Swagger	Widely used documentation method and user interface for Web API documentations. Supports multiple versions of the OpenAPI Standard.
Telegraf	Telegraf is a plugin-driven server agent for collecting and sending metrics and events from databases, systems, and sensors.
Time Series Database	A time series database (TSDB) is a database optimized for storing and analysing time series such as sensor or telemetry data.
TypeScript	TypeScript is a JavaScript-based strictly typed programming language that provides improved tools at any level in comparison to plain JavaScript.

Acronyms

DDD	Domain driven design Glossary: Domain driven design
DTO	Data transfer object Glossary: Data transfer object
DoD	Definition of Done
Go	Golang Glossary: Golang
INS	Institute for Networked Solutions at the Eastern Switzerland University of Applied Sciences
JAGW	Japapeno API Gateway developed by the INS
K8s	Kubernetes Glossary: Kubernetes
KISS	Keep it simple stupid
OR	Object relation
ORM	Object-relational mapping Glossary: Object-relational mapping
OST	Short form for the Eastern Switzerland University of Applied Sciences
PR / MR	Pull Request / Merge Request Glossary: Merge Request
PoC	Proof of concept
RUP	Rational Unified Process Glossary: Rational Unified Process
S.O.L.I.D	S - Single-responsibility principle O - Open-closed principle L - Liskov substitution principle I - Interface segregation principle D - Dependency Inversion Principle
SR	Segment Routing
SR-App	Segment Routing Application
Stderr	Standard Error Stream Glossary: Standard Error Stream

Stdout	Standard Output Stream Glossary: Standard Output Stream
TSDB	Time series database Glossary: Time series database
YAGNI	You aren’t gonna need it

References

[1] J. Hauser and P. Schlumpf, “Bachelor thesis Green Routing: Spring Term 2021,” Bachelor thesis, Institute for network solutions, OST – University of Applied Sciences, Rapperswil, 2022.

[2] C. Larman, *Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development,* 3rd ed. Upper Saddle River, NJ: Pearson; Prentice Hall, 2005.

[3] REFSQ; International Working Conference on Requirements Engineering: Foundation for Software Quality, *Requirements engineering: foundation for software quality: 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, March 14-17, 2016 : proceedings*. Cham, s.l.: Springer International Publishing, 2016.

[4] A. Wiggins, *The Twelve-Factor App.* [Online]. Available: https://12factor.net/ (accessed: Sep. 29 2021).

[5] J. Hauser and P. Schlumpf, “Term project Green Routing: Autumn Term 2021,” Term project, Institute for network solutions, OST – University of Applied Sciences, Rapperswil, 2021.

[6] D. Vincent, *A successful Git branching model.* [Online]. Available: https://nvie.com/posts/a-successful-git-branching-model/ (accessed: Oct. 12 2021).

[7] T. Preston-Werner, *Semantic Versioning 2.0.0.* [Online]. Available: https://semver.org/ (accessed: Jun. 21 2022).

[8] *Conventional Commits.* [Online]. Available: https://www.conventionalcommits.org/en/v1.0.0/ (accessed: Oct. 12 2021).

[9] *Prettier.* [Online]. Available: https://prettier.io/ (accessed: Jun. 21 2022).

[10] OpenJS Foundation, *ESLint.* [Online]. Available: https://eslint.org/ (accessed: Jun. 21 2022).

[11] Raphael 'kena' Poss, *Go (Golang) conding guidelines.* [Online]. Available: https://wiki.crdb.io/wiki/spaces/CRDB/pages/181371303/Go+Golang+coding+guidelines (accessed: Nov. 1 2021).

[12] *Golangci-lint is a Go linters aggregator.* [Online]. Available: https://golangci-lint.run/ (accessed: Oct. 31 2021).

[13] *Code formatting and naming convention tools in Golang.* [Online]. Available: https://www.golangprograms.com/code-formatting-and-naming-conventions-in-golang.html (accessed: Oct. 6 2021).

[14] Stretchr, Inc., *Testify - Thou Shalt Write Tests.* [Online]. Available: https://github.com/stretchr/testify (accessed: Oct. 31 2021).

C. Appendix

Change history

Version	Date	Changes	Responsible
1.0	24.06.2022	Finished the final appendix.	Jonas H. and Pascal S.

Contents

1. System test

During the end of the projects construction phase, the system was tested manually using the open-source API client Insomnia[9]. Requests templates for all API endpoints and methods, which are entirely suitable for system test requirements, were continuously produced during development. For the frontend, manual tests are performed directly on the website for all general functionalities.

The test requires that the whole Jalapeño system, particularly the Jalapeño API Gateway request service, is operational and that the network data is consistent. A system test is not possible without this perquisite.

The test was run on one of the developers powerful machines. The program performed exactly as it would have if it had been installed on a real server and connected to the Jalapeño API Gateway in the provided INS virtual Lab. Non-functional requirement tests, on the other hand, were performed using a large collection of mock data supplied from a network specification file. Because no real data was available at the time this system test took place, all telemetry data for the power consumptions and throughputs of the nodes was mocked.

1.1 For functional requirements

All system tests for functional requirements are based on the use cases specified in the requirements definition (part B).

1.1.1 Backend

1.1.1.1 Jalapeño data

ID	Request	Expected result	Fulfilled
T-F-1	GET /jalapeno/sync	Base tests are JD-1-A up to E. Processing and storing data: • Calculating average power consumption for every node • Syncing throughput for every node • Setting node links based on node edges • Storing all received and processed data into the database **Http status code**: 200 **Body**: empty	Yes
T-F-1-A	GET /jalapeno/nodes	Receiving all nodes from Jalapeño **Http status code**: 200 **Body**: nodes array	Yes

[9] Website: https://insomnia.rest/

T-F-1-B	GET /jalapeno/links	Receiving all links from Jalapeño **Http status code**: 200 **Body**: links array	Yes
T-F-1-C	GET /jalapeno/node-edges	Receiving all node edges from Jalapeño **Http status code**: 200 **Body**: node edges array	Yes
T-F-1-D	GET /jalapeno/telemetry/power/*:name* Parameter *name* with a specific node name identifier	Receiving power consumption telemetry data from Jalapeño about the specified node **Http status code**: 200 **Body**: telemetry data	Yes
T-F-1-E	GET /jalapeno/telemetry/throughput/*:name* Parameter *name* with a specific node name identifier	Receiving throughput telemetry data from Jalapeño about the specified node **Http status code**: 200 **Body**: telemetry data	Yes
T-F-2	GET /jalapeno/prefixes	Receiving all prefixes from Jalapeño **Http status code**: 200 **Body**: prefixes array	Yes
T-F-3	GET /jalapeno/srv6-sids	Receiving all SRv6 SIDs from Jalapeño **Http status code**: 200 **Body**: SRv6 SIDs array	Yes

1.1.1.2 Green routes

ID	Request	Expected result	Fulfilled
T-F-4	GET /green-routes	All previously calculated green routes for all nodes available **Http status code**: 200 **Body**: green routes array	Yes
T-F-5	GET /green-routes?*:nodeKey* Query *nodeKey* with a specific node key identifier	All previously calculated green routes for the specified node (ingress or egress) **Http status code**: 200 **Body**: green routes array	Yes
T-F-6	GET /green-routes/*:id*	Get calculated green route by a specific identifier	Yes

	Parameter *id* with a specific previously calculated green route identifier	**Http status code**: 200 **Body**: green route	
T-F-7	POST /green-routes/calculate/ *:ingressNodeKey*/*:egressNodeKey* Parameters with specific node key identifiers	Syncing latest telemetry data, calculating the greenest route and generating and deploying the SR-Policy accordingly. The green route must be calculated based on the currently activated green metrics. **Http status code**: 200 **Body**: green route	Yes
T-F-8	DELETE /green-routes/*:id* Parameter *id* with a specific previously calculated green route identifier	Deleting a previously calculated green route by id with the associated SR-Policy (including deprovisioning of the SR-Policy) **Http status code**: 200 **Body**: green route	Yes

1.1.1.3 Fastest routes

ID	**Request**	**Expected result**	**Fulfilled**
T-F-9	GET /fastest-routes	All previously calculated fastest routes for all nodes available **Http status code**: 200 **Body**: green routes array	Yes
T-F-10	GET /fastest-routes?*:nodeKey* Query *nodeKey* with a specific node key identifier	All previously calculated fastest routes for the specific node (ingress or egress) **Http status code**: 200 **Body**: green routes array	Yes
T-F-11	GET /fastest-routes/*:id* Parameter *id* with a specific previously calculated fastest route identifier	Get calculated fastest route by a specific identifier **Http status code**: 200 **Body**: green route	Yes
T-F-12	POST /fastest-routes/calculate/ *:ingressNodeKey*/*:egressNodeKey* Parameters with specific node key identifiers	Calculating the fastest route. **Http status code**: 200 **Body**: green route	Yes

1.1.1.4 Statistical purposes

ID	Request	Expected result	Fulfilled
T-F-13	GET /nodes	All previously received nodes from Jalapeño with a power consumption and throughput history **Http status code**: 200 **Body**: nodes array	Yes
T-F-14	GET /nodes/*:nodeKey* Parameter *nodeKey* for a specific node key identifier	Get received node from Jalapeño with a power consumption and throughput history **Http status code**: 200 **Body**: node	Yes
T-F-15	DELETE /nodes/*:nodeKey* Parameter *nodeKey* for a specific node key identifier	Deleting a node by key. This includes all associated data linked (green routes, fastest routes, SR-policies and logical links) **Http status code**: 200 **Body**: node	Yes

1.1.1.5 Green metrics

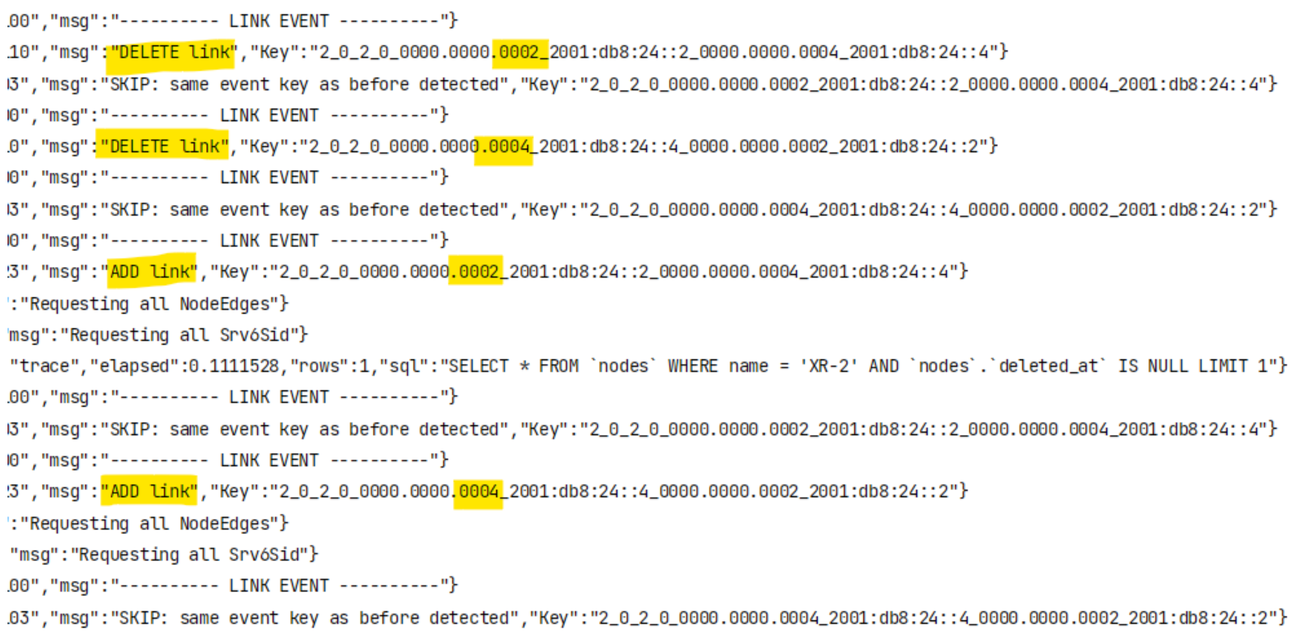
ID	Request	Expected result	Fulfilled
T-F-16	GET /green-metric-types	Receiving all green metrics types **Http status code**: 200 **Body**: green metric types array	Yes
T-F-17	PATCH /green-metric-types Body: array with *id* of the green metric type and *activated* value to be updated	Patching the activated state of all some or all green metric types **Http status code**: 200 **Body**: green metric types array	Yes

1.1.1.6 SR-Policies

ID	Request	Expected result	Fulfilled
T-F-18	GET /sr-policies	Receiving all generated SR-Policies **Http status code**: 200 **Body**: SR-policies array	Yes
T-F-19	GET /sr-policies *?greenRouteId* Query *greenRouteId* for a specific green route identifier	Receiving all generated SR-Policies for a specific green route **Http status code**: 200 **Body**: SR-policies array	Yes
T-F-20	GET /sr-policies/*:id* Parameter *id* with a specific previously generated SR-Policy identifier	Get generated SR-Policy by its id **Http status code**: 200 **Body**: SR-policy	Yes
T-F-21	GET /sr-policies/generate/ *:greenRouteId* Parameter *greenRouteId* with a specific previously calculated green route identifier	Generating SR-Policies for a specific previously calculated green route **Http status code**: 200 **Body**: SR-policies array	
T-F-22	GET /sr-policies/deploy/ *:srPolicyId* Parameter *srPolicyId* with a specific previously generated SR-Policy identifier	Deploying a previously generated SR-Policy **Http status code**: 200 **Body**: SR-policy	Yes
T-F-23	DELETE /sr-policies/*:srPolicyId* Parameter *srPolicyId* with a specific previously generated SR-Policy identifier	Deleting a SR-Policy by id including the deprovisioning in the network **Http status code**: 200 **Body**: SR-policy	Yes

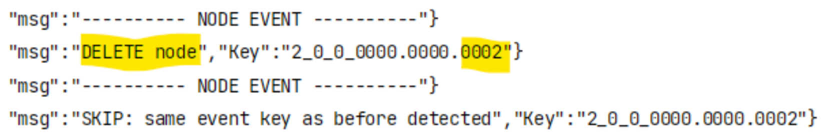
1.1.1.7 Jalapeño live subscription

The live network subscription must be able to react to link events. In the image below the logs are shown, which demonstrate the correct processing of these events by JAGW. In order to trigger these events, a link was previously dropped on the network.



```
.00","msg":"---------- LINK EVENT ----------"}
.10","msg":"DELETE link","Key":"2_0_2_0_0000.0000.0002_2001:db8:24::2_0000.0000.0004_2001:db8:24::4"}
03","msg":"SKIP: same event key as before detected","Key":"2_0_2_0_0000.0000.0002_2001:db8:24::2_0000.0000.0004_2001:db8:24::4"}
00","msg":"---------- LINK EVENT ----------"}
.0","msg":"DELETE link","Key":"2_0_2_0_0000.0000.0004_2001:db8:24::4_0000.0000.0002_2001:db8:24::2"}
00","msg":"---------- LINK EVENT ----------"}
03","msg":"SKIP: same event key as before detected","Key":"2_0_2_0_0000.0000.0004_2001:db8:24::4_0000.0000.0002_2001:db8:24::2"}
00","msg":"---------- LINK EVENT ----------"}
3","msg":"ADD link","Key":"2_0_2_0_0000.0000.0002_2001:db8:24::2_0000.0000.0004_2001:db8:24::4"}
':"Requesting all NodeEdges"}
msg":"Requesting all Srv6Sid"}
"trace","elapsed":0.1111528,"rows":1,"sql":"SELECT * FROM `nodes` WHERE name = 'XR-2' AND `nodes`.`deleted_at` IS NULL LIMIT 1"}
.00","msg":"---------- LINK EVENT ----------"}
03","msg":"SKIP: same event key as before detected","Key":"2_0_2_0_0000.0000.0002_2001:db8:24::2_0000.0000.0004_2001:db8:24::4"}
00","msg":"---------- LINK EVENT ----------"}
3","msg":"ADD link","Key":"2_0_2_0_0000.0000.0004_2001:db8:24::4_0000.0000.0002_2001:db8:24::2"}
':"Requesting all NodeEdges"}
"msg":"Requesting all Srv6Sid"}
.00","msg":"---------- LINK EVENT ----------"}
.03","msg":"SKIP: same event key as before detected","Key":"2_0_2_0_0000.0000.0004_2001:db8:24::4_0000.0000.0002_2001:db8:24::2"}
```

In addition, node events must also be able to be processed. As with the links, the logs are shown in the next image to demonstrate that they are working correctly. . In order to trigger these events, a node was previously shut down on the network.



```
"msg":"---------- NODE EVENT ----------"}
"msg":"DELETE node","Key":"2_0_0_0000.0000.0002"}
"msg":"---------- NODE EVENT ----------"}
"msg":"SKIP: same event key as before detected","Key":"2_0_0_0000.0000.0002"}
```

Since no events are generated on JAGW for node edges, they are processed together with the link events per request.
In addition, two other errors on the part of JAGW play a role here. Firstly, the API Gateway always closes the connection after five minutes and secondly, the cache of the JAGW has a data inconsistency, which can also lead to a crash of the Green SR-App processing service. Both errors are unfortunately irreparable on the side of the Green SR-App and therefore the live subscription can be deactivated.

1.1.1.8 Router configuration

To validate the successful configuration of the green route on the necessary routers in the network, the two figures below display the relevant configuration snippet for each case.

Ingress router

```
segment-routing
 traffic-eng
  segment-list usid_list_1
   index 10 srv6 usid 2222:2
   index 20 srv6 usid 7777:7
   index 30 srv6 usid 8888:8
   !
  !
 !
policy policy-1
   srv6
    locator 1to8 binding-sid dynamic behavior ub6-insert-reduced
   !
   color 2 end-point ipv6 2001:db8:8888::8
   candidate-paths
    preference 10
     explicit segment-list usid_list_1
     !
    !
   !
  !
srv6
  locators
   locator 1to8
    micro-segment behavior unode psp-usd
    prefix 2001:0:8::/48
   !
  !
```

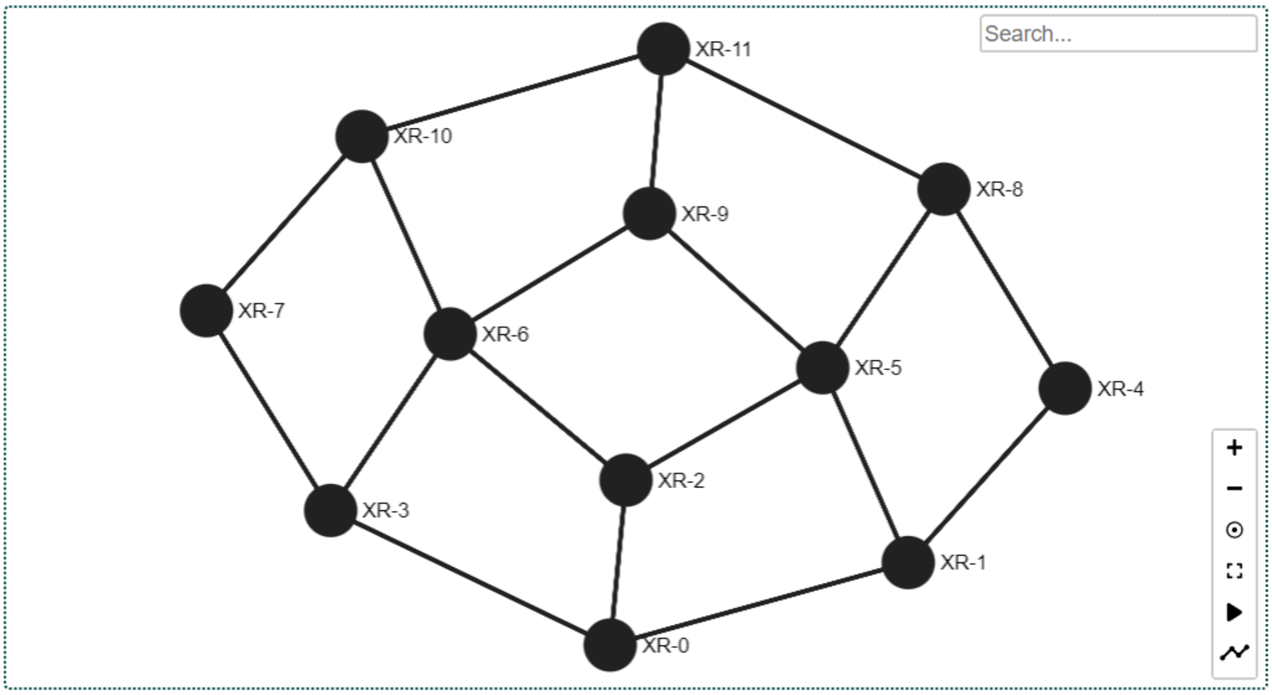
Egress router

```
prefix-set policy-11
  172.16.96.128/32
end-set
!
prefix-set policy-12
  172.16.96.0/32
end-set
!
extcommunity-set opaque 2:1
  2
end-set
!
extcommunity-set opaque 2:2
  3
end-set
!
route-policy SET_POLICY_COLORS_Cust-0
  if destination in 2:1 then
    set extcommunity color 2:1
  elseif destination in 2:2 then
    set extcommunity color 2:2
  endif
  pass
end-policy
!
vrf VPN1
  rd auto
  address-family ipv4 unicast
   export route_policy SET_POLICY_COLORS_Cust-0
    rd 1111:0
   !
   segment-routing srv6
    alloc mode per-vrf
   !
  !
```

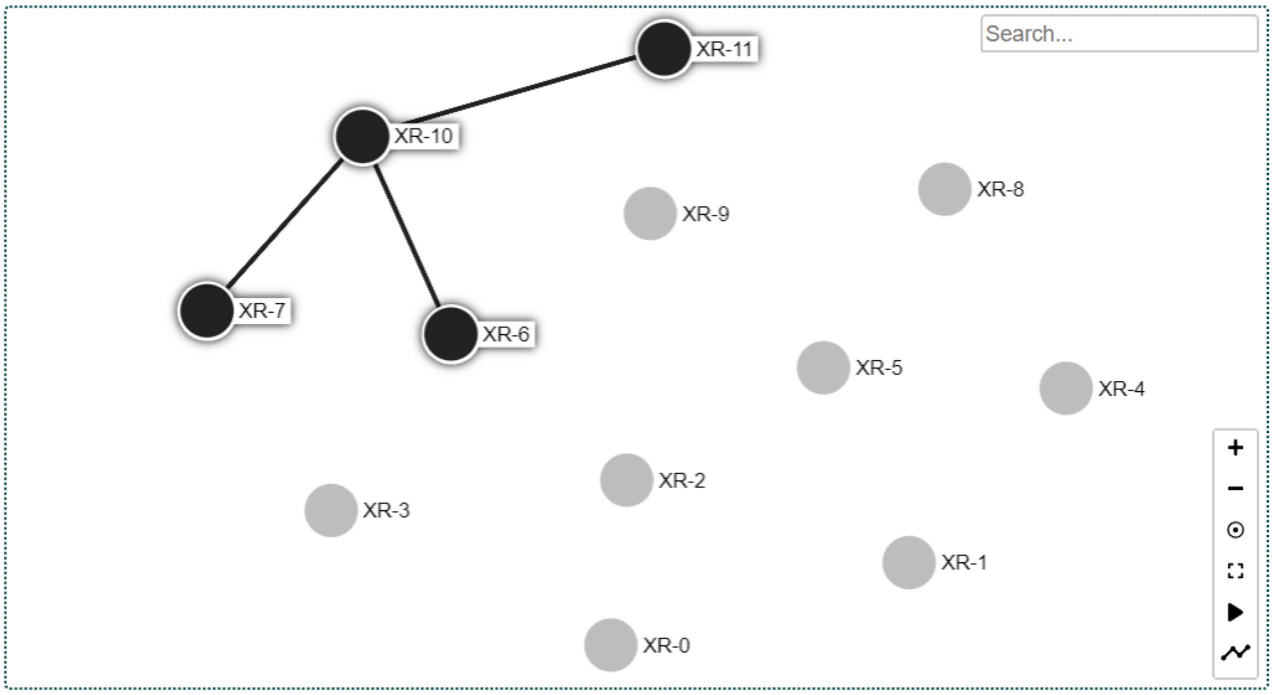
1.1.2 Frontend

1.1.2.1 Display network graph

It is possible to display the network graph. The nodes are arranged in the best possible way using different inbuilt algorithms from Simga.js. Additionally, nodes can be searched, and further options (at the bottom of the figure below) are available to interact with the graph.

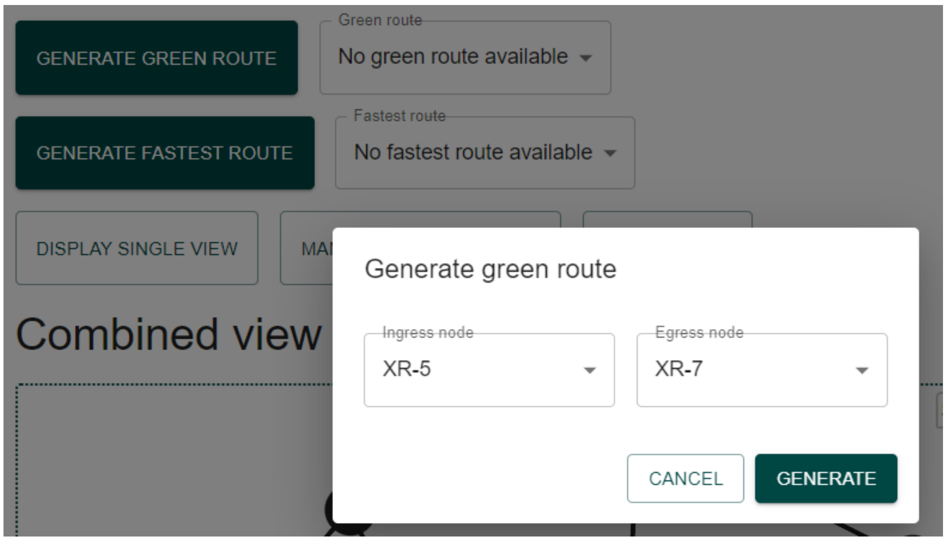


Furthermore, the links of a node with other nodes can be highlighted to get a better overview, especially for larger networks with many more links.



1.1.2.2 Generate routes

A green route can be generated using the button at the top left of the image below. Clicking on the button opens a pop-up where the generation can be started by specifying both the ingress and egress node and clicking the respective button.

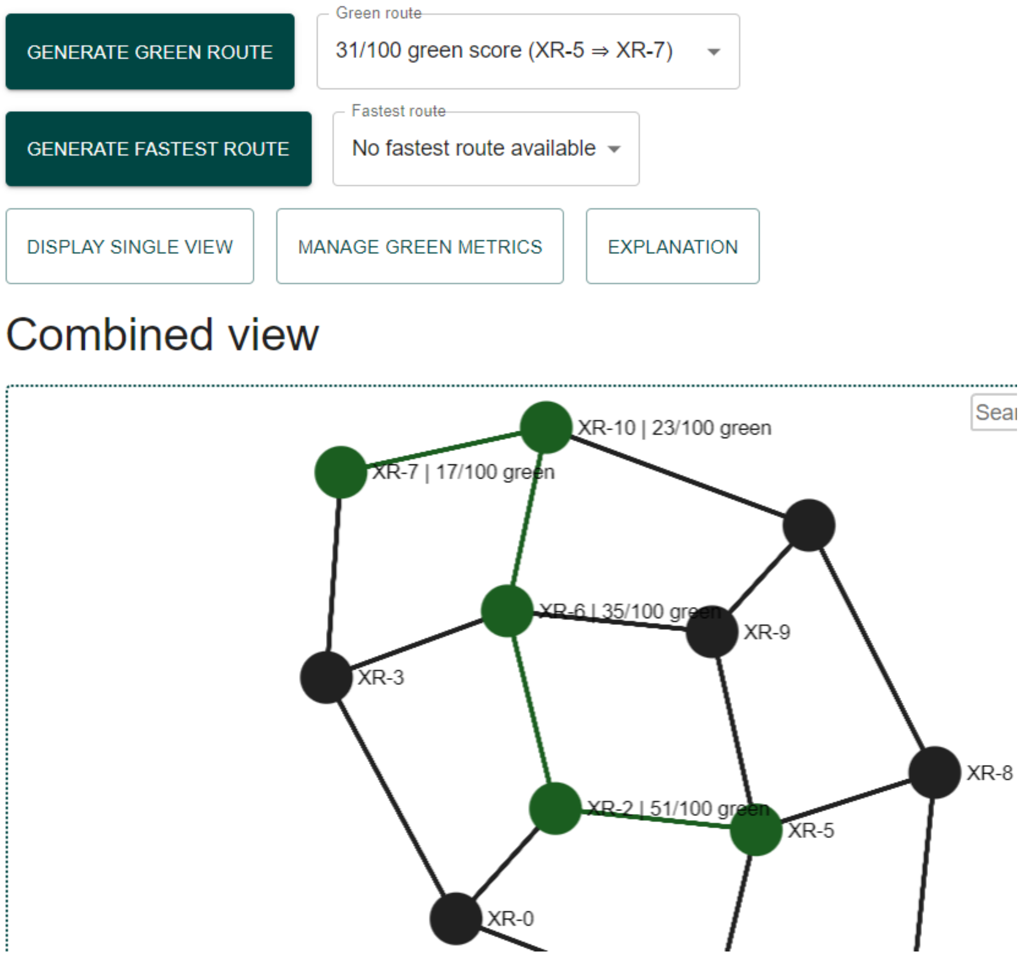


Generate green route

Ingress node: XR-5

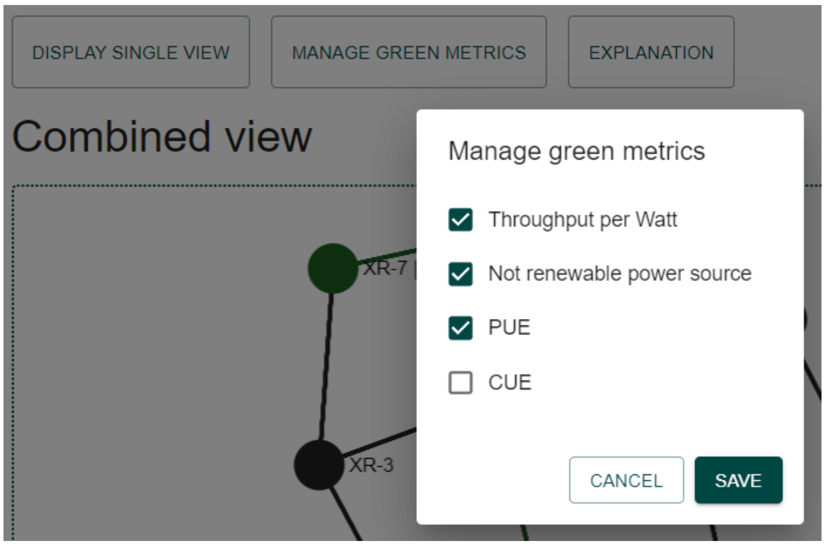
Egress node: XR-7

After the successful generation of a new green route, the path with all used metrics is automatically displayed in the network graph. In the dropdown next to the button for generating the route, the route is pre-selected, and the total green score is displayed, as visualized in the next figure.



1.1.2.3 Update green metrics

The button "Manage green metrics" opens a pop-up that displays all available green metrics and displays which of them are currently activated and which are deactivated. The activation status of the green metrics can also be adjusted via the pop-up.

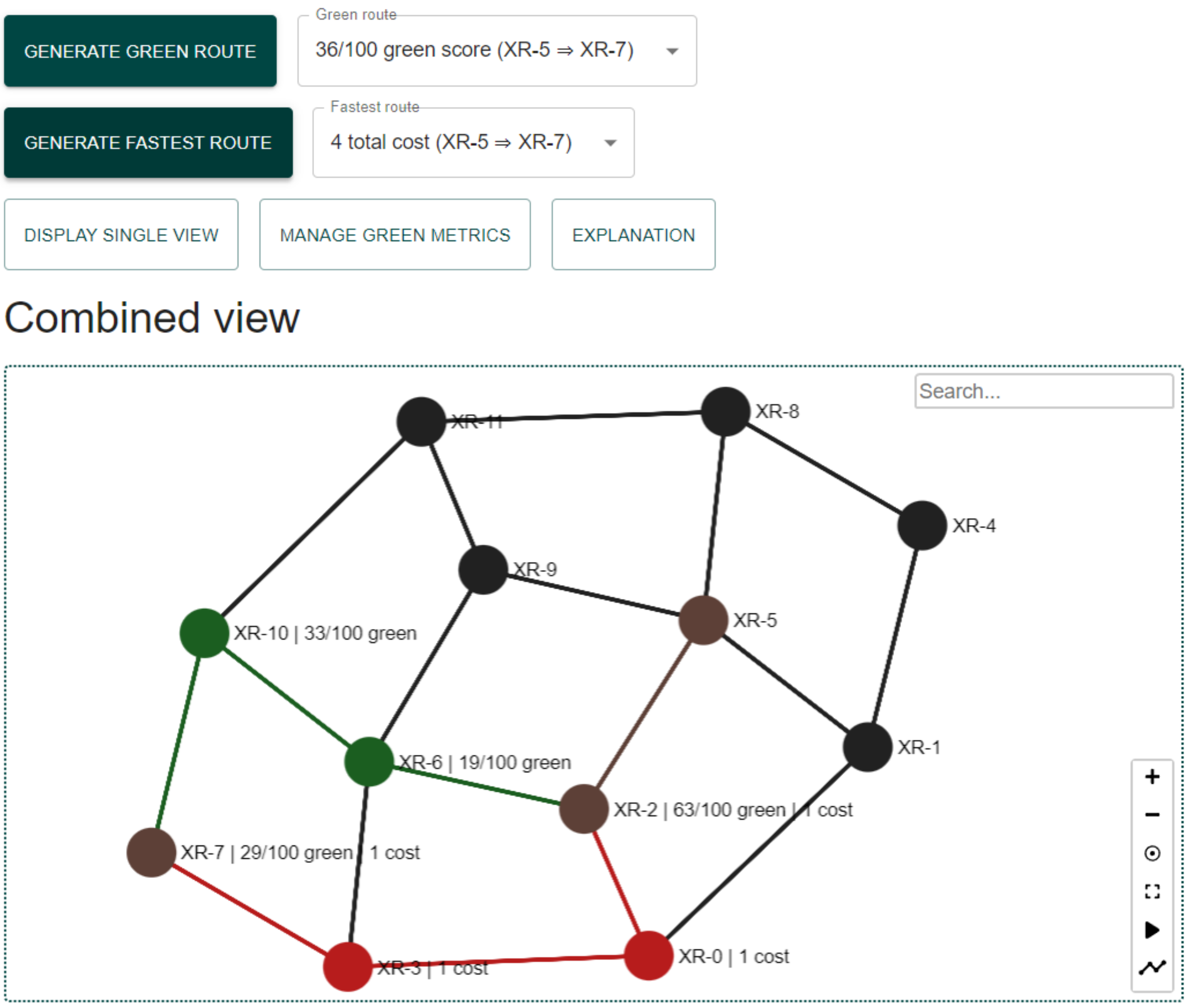


Manage green metrics

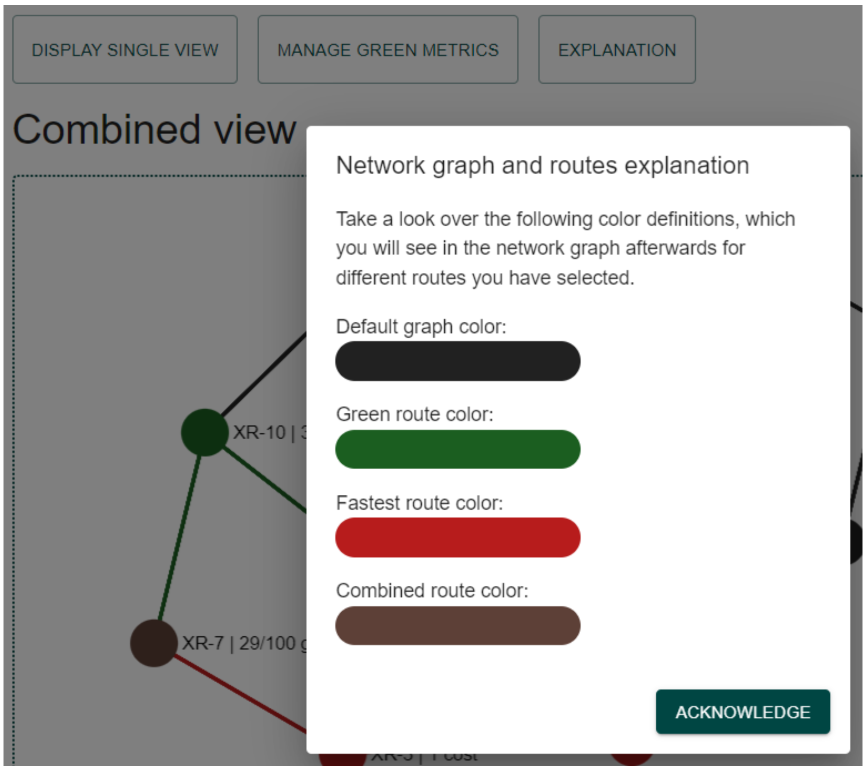
☑ Throughput per Watt
☑ Not renewable power source
☑ PUE
☐ CUE

1.1.2.4 Compare scenarios

The button in the upper left corner "Generate fastest route" can be used to generate the traditional fastest route for an ingress and egress node. This works in the frontend basically the same way as generating a new green route.
Through this option, the green route can be compared to the fastest route. Both paths are displayed in the network graph in a different color. Green stands for the green route, blue for the fastest route and overlaps are colored brown.


GENERATE GREEN ROUTE
Green route
36/100 green score (XR-5 ⇒ XR-7)
GENERATE FASTEST ROUTE
Fastest route
4 total cost (XR-5 ⇒ XR-7)
DISPLAY SINGLE VIEW
MANAGE GREEN METRICS
EXPLANATION
Combined view
Search...
XR-11
XR-8
XR-4
XR-9
XR-5
XR-10 | 33/100 green
XR-1
XR-6 | 19/100 green
XR-2 | 63/100 green | 1 cost
XR-7 | 29/100 green | 1 cost
XR-0 | 1 cost
XR-3 | 1 cost

There is also an "Explanation" button that opens a pop-up in which the colors used are explained.



Network graph and routes explanation

Take a look over the following color definitions, which you will see in the network graph afterwards for different routes you have selected.

Default graph color:

Green route color:

Fastest route color:

Combined route color:

ACKNOWLEDGE

1.2 For non-functional requirements

All system tests for non-functional requirements are based on the definition of non-functional requirements specified in the requirements definition (part B).

1.2.1 Functionality

Based on the definition for this non-functional requirement, no tests are required for the topic's security and interoperability.

1.2.2 Usability

ID	Based on	Expected behaviour and possibilities	Fulfilled
T-NF-1	Understandability	The green route and node objects provide data about the power consumption, throughput and other manually defined green metrics for links between nodes involved and a resulting cumulated green score for the whole route. It is possible to view a history for all green scores of green routes and node power consumptions and throughputs for each node.	Yes
T-NF-2	Operability	No possibility to test this completely, but the network configurations worked without problems on the	(Yes)

network until now. It worked until now without issues to provision and deprovision configurations on routers.

1.2.3 Reliability

ID	Based on	Expected behaviour and possibilities	Fulfilled
T-NF-3	Availability	Since the demands for this are not high, no special deployment setup was implemented to reach a very high availability. However, if the frontend or backend should crash during runtime, they will be automatically restarted by Kubernetes.	Yes
T-NF-4	Recoverability	The application system is developed according to the 12-Factor Methodology to ensure that it meets the Cloud Native Standard. Our app passed all 12 factors, with only one small discrepancy as described in the documentation. It is very easy to redeploy the Green SR-App and it automatically restarts itself after a crash.	Yes

1.2.4 Performance

ID	Based on	Expected behaviour and possibilities	Fulfilled
T-NF-5	Capacity	The application can handle a green route calculation for a network with 1000 nodes and many times more links in between without any problems or many resources.	Yes
T-NF-6	Time behavior	In less than 10 seconds, the program should calculate a green route and deploy the SR-Policy to the network. With 1000 nodes on fast desktop PCs inside the docker build environment, the achieved value is under 1 second, but without the deployment of the SR-Policy. The SR-Policy can only be deployed in the real data mode (no mocked data). While connected to the INS virtual Lab with no mocked data, the complete calculation and deployment of the SR-Policy takes around 8 seconds. The Jalapeño API Gateway must be functioning in normal workload with typical response times as a precondition.	Yes

1.2.5 Scalability

Since the demands for this are not high due to the purposes for demonstrations only, no special deployment setup was implemented to automatically scale the running application. However, this can be achieved relatively easy by increasing the count of replicas of the system on Kubernetes.

1.2.6 Maintainability

ID	Based on	Expected behavior and possibilities	Fulfilled
T-NF-7	Analyzability	Multiple log levels must be available in the backend, which can be controlled by environment variables. The log events vary according to the stated and accessible options.	Yes

2. Meeting minutes

24.02.2022, Project kick-off

Participants

- Prof. Laurent Metzger
- Michel Bongrad
- Jonas Hauser
- Pascal Schlumpf

Agenda and notes

1. BA Planung
    a. Idee: Selbstorganisierte Planung durch uns
        i. Oder doch noch ein Workshop? Nur noch nächste Woche für uns möglich (Planung)
    b. Ausarbeitung Projektplan diese und nächste Woche
    c. Vorstellung Zeitplan Draft
    d. Priorisierter Product Backlog (noch in Arbeit)
    e. Streng iterativ mit vollwertigen Ergebnissen nach jedem Sprint
        i. Ausnahme Sprint 1 für Inception
2. Server und Services
    a. Gleiches Setup wie in SA geplant
        i. MS Teams, YouTrack, GitLab, CI/CD mit INS Runners
        ii. GoLand, Docker, Kaniko
    b. Zusatz K8S Deployment
        i. Yannick Zwicker
    c. SonarQube geht nicht mehr
        i. Bei Yannick Zwicker melden
    d. Ob weiteres nicht mehr funktioniert, wissen wir jetzt noch nicht
    e. Wie ist der Stand um die Daten von den Routern zu bekommen?
        i. Wurde nicht weiter verfolgt seit Weihnachten
        ii. Michel hat einen Workaround gebaut -> muss noch getestet werden
        iii. Michel wird sich dem Thema wieder annehmen
        iv. Fokus auf Mocken/Faken, mehrere Möglichkeiten für die Umsetzung
            1. Direkt in kafka topic schreiben
            2. Direkt in InfluxDB schreiben
            3. Generierte Meldungen an Telegraf senden (wie bisher)
3. Aktueller Arbeitsfortschritt
    a. Systeme / Tools aufsetzen
    b. Dokumentationstemplate
    c. Projektplan

03.03.2022, Weekly exchange

Participants

- Prof. Laurent Metzger
- Julian Klaiber
- Michel Bongrad
- Severin Dellsperger
- Jonas Hauser
- Pascal Schlumpf

Agenda

1. Brainstorming Product Backlog (10 bis 11 Uhr)
    a. Erweiterung der Requirements aus SA
        i. UC07 Login löschen?
        ii. UC06 View routes wird nur umgesetzt durch Anbindung an Frontend vom INS (optional)
        iii. Actors beibehalten? Authentifizierung & Autorisierung nach uns nicht nötig (Gibt wichtigeres umzusetzen)
2. Verschiben: BA Erwartungen / allgemeine Anforderung (von Laurent)
3. Projekt Plan Review (Plan ab 11 Uhr)
    a. Wichtig: Zeitplan, Meilensteine, Risiko mgmt
4. Konkrete Mängel in der SA Dokumentation, die wir verbessern können?
5. Updates deployment auf unseren Server sr-000166 (Michel)
6. Aktueller Arbeitsfortschritt
    a. Development concept draft done
    b. Requirements wip
7. Outlook
    a. M1 abschliessen (Projekt Plan)
    b. M2 WIP (Requirements)
    c. Start Elaboration (Research and conceptions)
8. Pascal und Jonas abwesend nächsten Donnerstag (HSR Skitag)
    a. Wir senden die definierten Requirements frühzeitig per Teams und erwarten gerne bis zum Wochenende die Bestätigung / das Feedback

Notes on each agenda item discussed

1. Notes
    a. SR Aspekt ist wirklich Prio 1 für die Umsetzung aus Sicht von Laurent
        i. Muss auf SRv6 umgesetzt werden, mit IPv6 Adressen
        ii. Gibt eine Einführung am Mittwoch 09.03 15:00-17:00 in SR
        iii. Das Team schaut ob sie das Netzwerk neu deployen wollen auf eine neuere Version des Images, dieses könnte die Konfiguration erleichtern
    b. Alle Einverstanden mit der Umstellung auf Subscription
    c. Metrik g CO_2 /Mbit wäre grundsätzlich gut. Man muss die Stromart pro Rechenzentrum nehmen und als Konfiguration pro Node nehmen
        i. Gut wäre wenn man aus mehreren Pfaden auswählen könnte
        ii. Throughput wird fix als Metrik genommen
        iii. Carbon footprint monitoring anschauen
        iv. Vergleich von Greenest Path zu Shortest Path mit der Einsparung über einen Zeitraum
        v. Average verwenden für einen stabilen Pfad aber trotzdem auch die Realtimedaten berücksichtigen
    d. Fokus soll auf einer Demo liegen welche die Möglichkeiten aufzeigt die man im Bereich SR und Ökologie machen kann
    e. Pipeline optimierung ist eher Nebenthema
    f. Deployment auf Kubernetes wird umgesetzt und wird geschätzt
    g. Es wird ein einfaches GUI gewünscht zu Präsentationszwecken um auch sehen zu können was passiert
        i. Optimalerweise werden auch die Pfade angezeigt.
        ii. Man sollte auch den Unterschied von Fastest path zu greenest path sehen
2. Zugriff auf Swisscom Netz ist möglich aber nicht hilfreich aktuell
    a. Möglicherweise gibt es noch andere Router die man nutzen könnte im IATF-Netz, ist nicht für uns Relevant

3. Wird nächste Woche beim persönlichen Treffen besprochen basierend auf den heute definierten Zielen
4. Zwischenpräsentation in der Kalenderwoche 19 inklusive Michel
    a. Schlusspräsentation in der KW 25 inklusive Michel
5. Michel wird die SA Doku durchschauen und Feedback geben
6. Jalapeño hat ein Update bekommen und es hat ein paar Änderungen am Gateway gebraucht. Michel wird es am Nachmittag deployen auf dem Server
    a. Support und eventuelle Weiterentwicklung sind neu bei Dominique Illi und Urs …
7. Ok
8. Meeting verschoben auf Mittwoch ab 15:00

10.03.2022, Weekly exchange

Participants
- Prof. Laurent Metzger
- Michel Bongrad
- Severin Dellsperger
- Jonas Hauser
- Pascal Schlumpf

Agenda

1. Requirements in Arbeit
    a. Wird noch diese Woche fertig gestellt
2. BA Erwartungen / allgemeine Anforderung (von Laurent)
3. Segment routing introduction from Laurent
4. Feedback zur SA Dokumentation (von Michel evtl.)
5. Simulate network changes for live updates (jagw subscriptions)
6. Aktueller Arbeitsfortschritt
    a. Development concept
    b. Requirements
7. Outlook
    a. M1 abschliessen (Projekt Plan)
    b. M2 WIP (Requirements)
    c. Start Elaboration (Research and conceptions)

Notes on each agenda item discussed

1. Neue IOS-XR Image Version ist 7.5.1 und diese wird auch im INS virtual Lab provisioniert
2. Wichtige Punkte
    a. Mindestens drei Faktoren für die Metriken
    b. Deployment auf K8s Cluster
    c. Scenarios für die Demo
        i. Shortest Path vs Greenest Path
        ii. Demo mit Iperf auf realem netz
        iii. Für grosse Netzwerke mit gemockten Daten ohne Iperf
        iv. Stromdaten werden in einem Register pro Router gesammelt
3. Nichts anzumerken
4. Feedback
    a. Bilder zentriert, leere Zeilen vorher und nacher
    b. Bei Bildern immer Quelle angeben auch wenn Eigenkreation

   c. Teil B in den Anhang
   d. List of Figures etc. ebenfalls Nummerieren
   e. Nicht mehr als 4 Verschachtlungsebenen
   f. Keine Bilder oder Tabellen auf die nicht verwiesen wird
   g. Erster Satz in einem Abschnitt macht klar um was es geht
   h. Letzter Satz in einem Abschnitt fasst alles noch einmal zusammen
5. Verschoben (bilateral besprechen)

17.03.2022, Weekly exchange

Participants

- Prof. Laurent Metzger
- Julian Klaiber
- Michel Bongrad
- Severin Dellsperger
- Jonas Hauser
- Pascal Schlumpf

Agenda

1. Feedback zu Requirements
2. JAGW fehlende Endpoints
   a. l3vpn_v4_prefix (content in Jalapeno: yes)
   b. ls_srv6_sid (content in Jalapeno: yes)
   c. Peer? (content in Jalapeno: yes)
   d. Neues Risiko wird erstellt
3. Inkonsistente Daten von Jalapeno
   a. Behindern bei der Arbeit
4. VRF Introduction (von Severin)
   a. Von wo kommen die Daten?
   b. Alle nötigen Daten im Domain Model?
5. Status Virtual Lab IOS-XR 7.5.1 upgrades
6. Info: Go v1.18 jetzt offiziell veröffentlicht
7. Aktueller Arbeitsfortschritt
   a. App für Segments bereit: Domain model v1
   b. Frontend PoC
   c. Green Index metrics
      i. Aktuell ca. 15 Ideen
      ii. Interview mit Swisscom
   d. Calculation algorithm
8. Outlook
   a. Green index
   b. Configure real network
9. Letzte Arbeitswoche von Michel beim INS

Notes on each agenda item discussed

1. Keine spezifischen Rückmeldungen zu den Anforderungen
2. Infos
   a. Prefix ist auch Interessant, Base attributes -> ext_community_list
   b. Vpn rd ebenfalls wichtig
   c. Es hat zu viele Abhängigkeiten um es schnell im Gateway zu implementieren

d. Evt direkte Verbindung zu Arango ohne Gateway
   i. Abhängig von der evtl. Weiterentwicklung
e. Direkt auf Kafka topic zugreifen um Änderungen mitzubekommen ist eine Alternative um zu sehen was sich geändert hat
f. Es wird am 17.03.22 Nachmittag einen Entscheid gefällt zur Weiterentwicklung des Gateway
g. Wenn bis am Abend keine andere Information kommt, wird das Topic nicht im Gateway sein

3. Es gibt einen Fix, welcher anscheinend funktioniert
   a. Wurde auf unserer Seite noch nicht angewendet
   b. Fix wird zeitnahe auf unserer Seite angewendet
4. Inputs von Severin
   a. VRF ist ein virtueller Space für Kundennetzwerke
   b. RouteDistinguisher fügt Tag hinzu welcher sagt zu welchem Netz es gehört
   c. Route targets könne tags exportieren welche von anderen VRFs importiert werden können
      i. Ermöglicht VRF übergreifende Sachen
      ii. Nur wenn export und import tag matchen funktioniert VRF
   d. 10.10.10.10:1 um Netz zu identifizieren als "Hack"
   e. In 99% der Fälle ist export und import tags identisch
      i. Da keine andere Möglichkeit muss davon ausgegangen werden, dass diese identisch sind
   f. RD manuell setzen und logik implementieren um vrf name zu generieren
   g. RD type immer 1 -> nicht beziehen
   h. https://packetlife.net/blog/2013/jun/10/route-distinguishers-and-route-targets/
5. Ist bereit und konfiguriert
6. Infos
7. Infos
   a. Rückmeldungen
      i. Node hat mehr einen locator als eine SegmentId, durch IPv6 gibt es mehr Felder
      ii. Nach der Destination auch Color exportieren
   b. Mit zwei unidirectional links gibt es probleme bei der Darstellung -> zu einem bidirectional link umwandeln
      i. Positionen von den Routern hardcoden und nicht dynamisch erstellen

31.03.2022, Weekly exchange

Participants

- Prof. Laurent Metzger
- Julian Klaiber
- Severin Dellsperger
- Jonas Hauser
- Pascal Schlumpf

Agenda

1. JAGW/Jalapeno Pain Points, gelistet nach Priorität (Liste an Dominique gesendet)
   a. LsNodeEdges Dateninkonsistenz (@Julian)
   b. Subscription stream timeout error nach exakt 5min
      i. Error: "rpc error: code = Unknown desc = stream timeout"
      ii. Lösungsvorschlag auf seitens Server (JAGW)
      iii. Idee "keepalive": https://github.com/grpc/grpc-go/issues/5059#issuecomment-994997711

    c. Keine events bei SubscribeToLsNodeEdges()
        i. Workaround seitens Green-SR App möglich, sofern Daten konsistent sind
    d. Fehlende Collections von Jalapeno im JAGW
        i. "l3vpn_v4_prefix"
        ii. "peer" (unbekannt, ob wirklich nötig)
2. Weiterer JAGW pain point bekannt
    a. Request service gibt auch Nodes und Links mit "del" status retour
    b. In Jalapeno sind diese aber nicht zu finden
    c. Evtl. ein Caching Problem?
3. Ausbau Wechsel auf Subscription on hold, weil zu viele pain points mit JAGW (funktioniert mit NodeEdges on demand)
    a. Aufwand und tatsächlicher Ertrag lohnt sich zum jetzigen Zeitpunkt nicht
    b. Fehlerbehandlung und Überprüfungen für Datenkonsistenz sind extrem aufwändig mit dem aktuellen Standpunkt
4. Aktueller Arbeitsfortschritt
    a. Alle dependencies up to date (exkl. Go v1.18)
    b. Neues Domain Model umgesetzt (mehrere neue Entities etc.)
    c. Direktverbindung zu Jalapeno (für z.b. l3vpnV4Prefix)
    d. Green Index handling and calculation
    e. JAGW subscription
5. Outlook
    a. API Ausbau
    b. Mocking und Testing
    c. Algorithmus v2 beta
    d. Get throughput from router
    e. Architektur Doku
    f. Abschluss M4

Notes on each agenda item discussed

1. JAGW Übersicht der Probleme
    a. Fix wird heute von Julian eingespielt auf Jalapeno, informiert wenn Wechsel vollzogen worden ist
    b. Kein Kommentar
    c. Julian schaut einmal in Kafka ob alle ok ist
    d. Sind nicht vorhanden im Gateway und wird wahrscheinlich nicht gelöst, da Workaround vorhanden
2. JAGW Part 2
    a. Liegt wahrscheinlich am Redis cache
3. Ist ok dass wir Subscription on hold stellen
4. Kein Kommentar
5. Bei verschiedenen Versionen ist es wichtig, die Gründe in der Doku zu beschreiben (bsp. Algorithmus)

14.04.2022, Weekly exchange

Participants

- Prof. Laurent Metzger
- Severin Dellsperger
- Jonas Hauser

Agenda

1. Teilausfall Pascal aufgrund Krankheit/Spitalaufenthalt
    a. Ca. eine Woche Arbeitszeitausfall
    b. Arbeit ab jetzt wieder und versucht die Zeit Schritt für Schritt einzuholen
    c. Leichte Verzögerung dadurch für die M4 Erreichung
2. JAGW/Jalapeno pain points:
    a. LsNodeEdges Dateninkonsistenz (Julian)
        i. Es scheint nach unserer App behoben zu sein
    b. Subscription stream timeout error nach exakt 5min (Dominique)
    c. Keine events bei SubscribeToLsNodeEdges() (Dominique)
    d. Fehlende Collections von Jalapeno im JAGW
        i. Workaround implementiert
    e. Request service gibt auch Nodes und Links mit "del" status retour, obwohl sie nicht mehr vorhanden sind in der ArangoDB von Jalapeno. (Dominique)
3. BA Zwischenpräsentation (Laurent)
    a. 18.05 15:00 oder 20.05 11:00
    b. SA PPT ausgehändigt an Laurent für Demo Zwecke
4. Welche Yang Models und Sensor Pfade nehmen um Throughput zu bekommen?
5. Aktueller Arbeitsfortschritt
    a. Jagw Subscription
    b. Get throughput from router
    c. Implementation LsPrefixes & LsSrv6Sid
    d. Algorithmus v2 beta
    e. API Ausbau (neue Endpoints)
6. Outlook
    a. Green route segments
    b. Generate SR-Policies
    c. Abschluss M4
    d. Start M5
        i. Configure network
        ii. Frontend
        iii. Deployment to K8s

Notes on each agenda item discussed

1. In Ordnung, sie wünschen Pascal gute Genesung
2. Details
    a. Problem sollte weiterhin bestehen, Severin klärt ab
    b. Dominique wird leider nicht sehr bald Zeit dafür haben
    c. Dasselbe wie zuvor
    d. Okey
    e. Dominique wird leider auch dafür nicht sehr bald Zeit haben, evtl. wird sich Severin darum kümmern
3. Um eine Woche verschoben
    a. Entweder 18.05 nach 15:00 oder
    b. 20.05 um 11:00
    c. Vor Ort in Rapperswil
4. Severin schaut es sich an und meldet sich
5. Kein Kommentar
6. Kein Kommentar

21.04.2022, Weekly exchange

Participants
- Prof. Laurent Metzger
- Julian Klaiber
- Severin Dellsperger
- Jonas Hauser
- Pascal Schlumpf

Agenda

1. BA Zwischenpräsentation (Laurent)
    a. 18.05 15:00 oder 20.05 11:00
2. Router Erkennung über angehängte Netzwerke nur über parsing möglich
    a. Collection L3_VPN_V4_Prefix
    b. Router-ID und Peer sind immer Reflektoren
    c. Erkennung welcher Router nur über next hop möglich
3. Updates zu JAGW pain points?
4. Aktueller Arbeitsfortschritt
    a. M4 abgeschlossen
    b. Start M5
5. Outlook
    a. Frontend part 1w
        i. Display graph
        ii. Generate Green route pop-up
        iii. Path highlighting
        iv. Etc.
    b. Configure network
        i. Generate SR-Policies
        ii. Config message
        iii. Green route CRUD network handling

Notes on each agenda item discussed

1. Definitiver Termin wird am 28.04 bestätigt
2. Julian schaut wie sie es gelöst haben auf ihrer JAGW Instanz und gibt Rückmeldung
3. Wir bis Ende BA nicht gelöst werden.
    a. Router Konfiguration soll vorbereitet werden
    b. Alle Probleme dokumentieren und laufend melden
4. Keine wichtigen Inputs
5. Keine wichtigen Inputs

28.04.2022, Weekly exchange

Participants
- Julian Klaiber
- Jonas Hauser

Agenda

1. Pascal wieder im Spital
    a. Macht eine zweiwöchige Pause in der BA
2. BA Zwischenpräsentation Termin
    a. 18.05 15:00 oder 20.05 11:00

3. Laurent zeigt Präsentation von letzter Woche
4. Aktueller Arbeitsfortschritt
    a. Frontend part 1w
        i. Graph dynamisch anzeigen
        ii. Generate Green route pop-up
        iii. Path highlighting basierend auf letzter berechneter Green Route
    b. Configure network
        i. Backned diverse Fehlerbehebungen
        ii. Generate SR-Policies
        iii. Protobuf repo Erstellung
5. Outlook
    a. Frontend Optimierungen
        i. Zweite Möglichkeit um bestehende Green Routes anzuzeigen
    b. Config message Generierung
    c. API Ausbau
    d. CRUD green route network handling
    e. BE Optimierung & Fehlerbehebungen

Notes on each agenda item discussed

1. Nach Besprechung mit Pascal möchte er eine Zweiwöchige Auszeit nehmen für die Genesung
    a. Julian kann keine definitive Aussage darüber treffen, wie wir weiter vorgehen sollen und wird darum dies mit Laurent bilateral besprechen
    b. Das Projekt verläuft allgemein trotzdem sehr gut und höchstwahrscheinlich müssen keine grösseren Umplanungen vorgenommen werden, sondern einfach gewisse weniger wichtigen Features weggelassen werden für die betroffenen Meilensteine bzw. Product Backlog items
    c. Laurent meldet sich bei uns
2. Traktandum verschoben, weil Laurent abwesend ist
3. Traktandum verschoben, weil Laurent abwesend ist
4. Ausgiebiger Austausch mit Julian
    a. Über das Frontend von ihrer BA inkl. Feedback (Demo aktueller Frontend Stand)
    b. Über das Backend und den Schwierigkeiten das Netzwerk zu konfigurieren und ein Handling zu machen für Green Route Änderungen (z.b. Green Route löschen)
        i. Info: Cisco ist daran eine Netzwerkprovisionierungs-API zu bauen, mit welcher die SR-Apps nicht mehr selbst die Netzwerke konfigurieren müssen. Diese API ist aber noch nicht fertig und kann darum noch nicht verwendet werden.
        ii. Potentiell alternatives Tool um Netzwerk zu konfigurieren: https://github.com/nornir-automation/gornir
5. Keine wichtigen Inputs

05.05.2022, Weekly exchange

Participants
- Prof. Laurent Metzger
- Julian Klaiber
- Severin Dellsperger
- Jonas Hauser
- Pascal Schlumpf

Agenda

1. Zweiwöchiger Ausfall von Pascal, weiteres Vorgehen
    a. Bis 15. Mai, Ende M5
    b. Anpassungen M5
        i. Routerkonfiguration abschluss verschieben
        ii. Deployment verschieben
2. BA Zwischenpräsentation Termin
    a. Definitiv am 20.05 um 11:00
3. Laurent zeigt Präsentation von vorletzter Woche
4. Anforderungen BA Zwischenpräsentation
5. Aktueller Arbeitsfortschritt
    a. Start M5 Sprint 2, nur halbe Zeit eingeplant
    b. Frontend
        i. Generate green route
        ii. Select and highlight green routes
        iii. Manage green metrics
    c. Router Konfiguration
        i. Beim Yang model für Traffic Engineering fehlt srv6 bei den Segmenten
        ii. Jetzt wird versucht es über SSH direkt zu konfigurieren
        iii. Gab noch Probleme mit SSH Standard Lib von Go
        iv. Wird noch versucht über gornir
6. Outlook
    a. Calculate fastes path und Vergleich im Frontend
    b. Green score besser anzeigen
    c. BE Bugfixes

Notes on each agenda item discussed

1. Es gibt zwei Varianten
    a. Verlängerung von 40h -> ist die präferierte Variante
    b. Oder Funktionsumfang reduzieren
    c. Entscheid gefallen auf Verlängerung
    d. Präsentation am 24.06 11:00 vor Ort
2. Keine weiteren Infos
3. Er stellt sie uns zur Verfügung
4. Eine Mischung aus Übersicht und technischen Aspekten sowie persönliche Erfahrungen und Herausforderungen
    a. Stefan Keller wird Gegenleser sein und auch an der Präsentation anwesend sein
    b. 20min (exkl. 5min Demo)
    c. Evtl. in Englisch, aber wenn alle Deutsch können dann in Deutsch
    d. Slides aber definitiv in Englisch
5. Bemerkungen
    a. Keine
    b. Wäre gut wenn ein Router ausgewählt werden könnte anstelle über die Keys
        i. Farbe des Pfades natürlich in Grün
        ii. Vergleich von schnellster zur Grünsten Route mit separaten Graphen ist besser
        iii. Ein Szenario mit 100 Routern und 500 - 1000 Links ist gewünscht

20.05.2022, Bachelor thesis interim presentation

Participants
- Prof. Laurent Metzger
- Prof. Stefan Keller

- Julian Klaiber
- Severin Dellsperger
- Jonas Hauser
- Pascal Schlumpf

Agenda

1. Feedback zur Präsentation

Notes on each agenda item discussed

2. Feedback zur Präsentation
    a. Green Routing in Networking erwähnen
    b. Grüne Metrik für Fastest Path auch anzeigen, um zu vergleichen
    c. IGP Metrik für Grüne Route anzeigen
    d. Grössere Topologie verwenden, einzelne Folie nur für den Vergleich nutzen
    e. Mehr Vergleich zwischen Routen in der Demo aufzeigen
    f. Direkt Green Score und IGP link metrik anzeigen
    g. Im Frontend evtl. Beide Graphen übereinanderlegen, um besser nachzuvollziehen
    h. Wenn nötig fixe Punkte (x/y) verwenden für die Graph Nodes, damit es besser aussieht
    i. Evtl. Ablaufdiagramm für den Kalkulierungsalgorithmus erstellen

09.06.2022, Weekly exchange

Participants

- Prof. Laurent Metzger
- Julian Klaiber
- Severin Dellsperger
- Pascal Schlumpf

Agenda

1. Genauer Termin für BA Präsentation
2. Aktueller Arbeitsfortschritt
    a. Frontend abgeschlossen
        i. Pfade könne nun kombiniert angezeigt werden
        ii. Man sieht nun sowohl die Link Kosten als auch den GreenScore auf den Edges
    b. Netzwerk config funktioniert aber konnte nicht kontrolliert werden
        i. Wie prüft man pfad von Packeten?
    c. Verschiedene Aufräumarbeiten und Behebung offener Punkte
3. Outlook
    a. Abschluss Entwicklung in diesem Sprint
    b. Abgabe Abstract und Poster
    c. Dokumentation fertigstellen
    d. Abgabe aller Teile

Notes on each agenda item discussed

1. Freitag 24.06.22 Vorort
2. Aktueller Arbeitsfortschritt
    a. Kein Kommentar
    b. Nur per debugging auf dem router möglich
        i. Nicht so wichtig

    c. Kein Kommentar
3. Outlook
    a. Laurent hat gefragt, ob wir auch bei der BA Präsentationen dabei sind am 17.06 -> Antwort Ja wir sind vor Ort am präsentieren

3. Project related configurations

3.1 Golangci-lint linters

- goconst
- gocritic
- gofmt
- gomnd
- Gocyclo
- goprintffuncname
- gosec
- gosimple
- govet
- bodyclose
- deadcode
- depguard
- dogsled
- errcheck
- errorlint
- exportloopref
- exhaustive
- ineffassign
- misspell
- nakedret
- prealloc
- predeclared
- revive
- staticcheck
- structcheck
- stylecheck
- thelper
- tparallel
- typecheck
- unconvert
- unparam
- unused
- varcheck