HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Bachelor Thesis

Streaming Telemetry Analytics

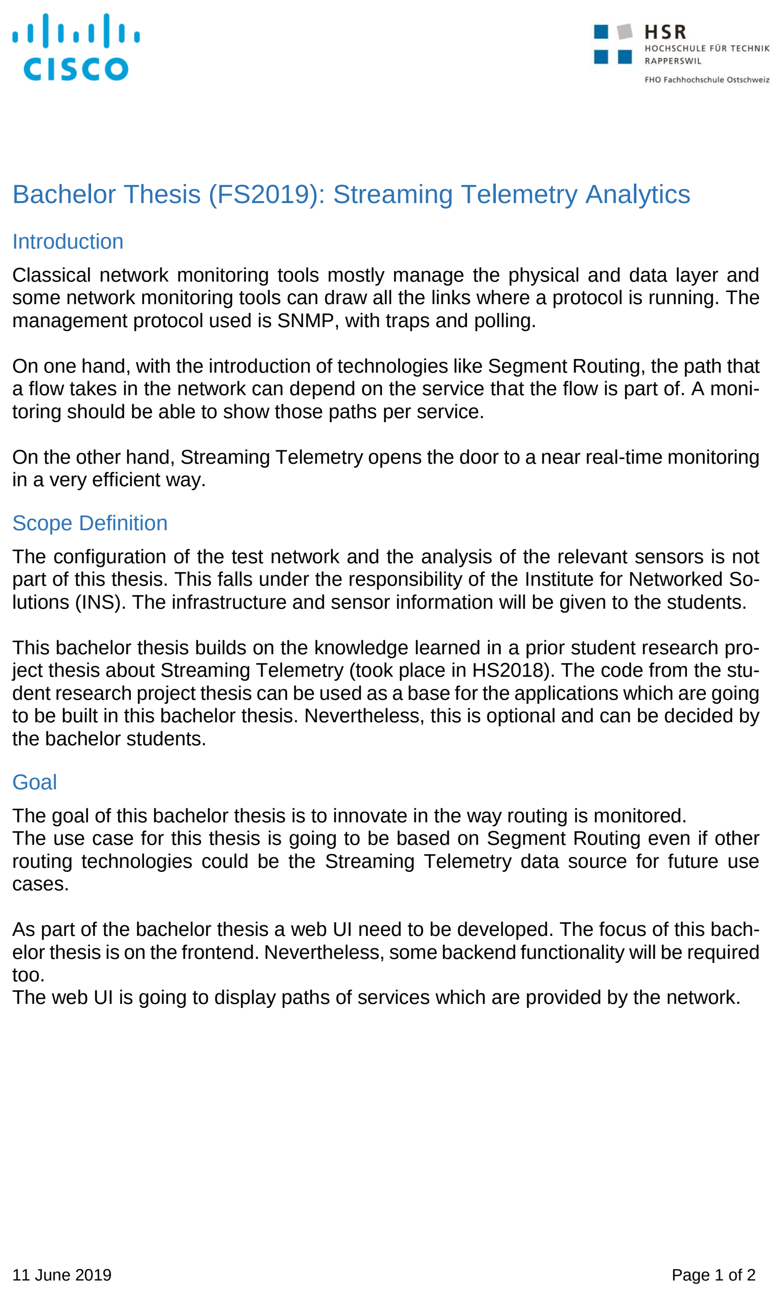
HSR Hochschule für Technik Rapperswil

Abteilung für Informatik

Period: 18.02.2019 - 14.06.2019

Authors	Faic Patrick	**Advisor**	Prof. Metzger Laurent
	Moosmann Benjamin	**Expert**	Billas Laurent
		Counterchecker	Richter Stefan

Assignment



Bachelor Thesis (FS2019): Streaming Telemetry Analytics

Introduction

Classical network monitoring tools mostly manage the physical and data layer and some network monitoring tools can draw all the links where a protocol is running. The management protocol used is SNMP, with traps and polling.

On one hand, with the introduction of technologies like Segment Routing, the path that a flow takes in the network can depend on the service that the flow is part of. A monitoring should be able to show those paths per service.

On the other hand, Streaming Telemetry opens the door to a near real-time monitoring in a very efficient way.

Scope Definition

The configuration of the test network and the analysis of the relevant sensors is not part of this thesis. This falls under the responsibility of the Institute for Networked Solutions (INS). The infrastructure and sensor information will be given to the students.

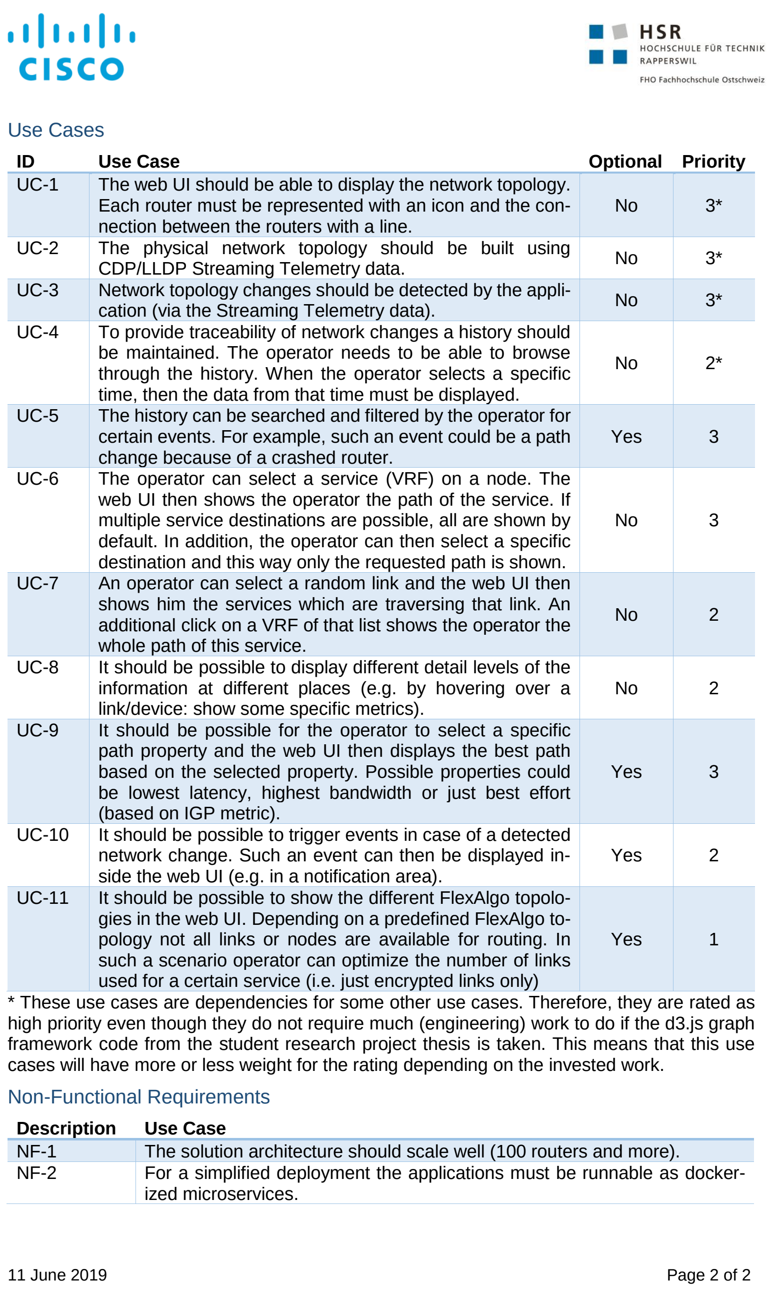
This bachelor thesis builds on the knowledge learned in a prior student research project thesis about Streaming Telemetry (took place in HS2018). The code from the student research project thesis can be used as a base for the applications which are going to be built in this bachelor thesis. Nevertheless, this is optional and can be decided by the bachelor students.

Goal

The goal of this bachelor thesis is to innovate in the way routing is monitored.
The use case for this thesis is going to be based on Segment Routing even if other routing technologies could be the Streaming Telemetry data source for future use cases.

As part of the bachelor thesis a web UI need to be developed. The focus of this bachelor thesis is on the frontend. Nevertheless, some backend functionality will be required too.
The web UI is going to display paths of services which are provided by the network.

11 June 2019 Page 1 of 2



Use Cases

ID	Use Case	Optional	Priority
UC-1	The web UI should be able to display the network topology. Each router must be represented with an icon and the connection between the routers with a line.	No	3*
UC-2	The physical network topology should be built using CDP/LLDP Streaming Telemetry data.	No	3*
UC-3	Network topology changes should be detected by the application (via the Streaming Telemetry data).	No	3*
UC-4	To provide traceability of network changes a history should be maintained. The operator needs to be able to browse through the history. When the operator selects a specific time, then the data from that time must be displayed.	No	2*
UC-5	The history can be searched and filtered by the operator for certain events. For example, such an event could be a path change because of a crashed router.	Yes	3
UC-6	The operator can select a service (VRF) on a node. The web UI then shows the operator the path of the service. If multiple service destinations are possible, all are shown by default. In addition, the operator can then select a specific destination and this way only the requested path is shown.	No	3
UC-7	An operator can select a random link and the web UI then shows him the services which are traversing that link. An additional click on a VRF of that list shows the operator the whole path of this service.	No	2
UC-8	It should be possible to display different detail levels of the information at different places (e.g. by hovering over a link/device: show some specific metrics).	No	2
UC-9	It should be possible for the operator to select a specific path property and the web UI then displays the best path based on the selected property. Possible properties could be lowest latency, highest bandwidth or just best effort (based on IGP metric).	Yes	3
UC-10	It should be possible to trigger events in case of a detected network change. Such an event can then be displayed inside the web UI (e.g. in a notification area).	Yes	2
UC-11	It should be possible to show the different FlexAlgo topologies in the web UI. Depending on a predefined FlexAlgo topology not all links or nodes are available for routing. In such a scenario operator can optimize the number of links used for a certain service (i.e. just encrypted links only)	Yes	1

* These use cases are dependencies for some other use cases. Therefore, they are rated as high priority even though they do not require much (engineering) work to do if the d3.js graph framework code from the student research project thesis is taken. This means that this use cases will have more or less weight for the rating depending on the invested work.

Non-Functional Requirements

Description	Use Case
NF-1	The solution architecture should scale well (100 routers and more).
NF-2	For a simplified deployment the applications must be runnable as dockerized microservices.

11 June 2019 Page 2 of 2

Abstract

For a network operator it is crucial to see the routing of data packets through a network. With the introduction of new technologies like Segment Routing, the path taken in the network can depend on the service and its assigned requirements (e.g. QoS, bandwidth, etc.). A modern monitoring tool should be able to show those paths per service in a dynamic way. Another technological innovation in the network domain is Streaming Telemetry. It opens the door to near real-time monitoring of a network in a very efficient way since its using a push and not a pull model like SNMP does.

The developed application facilitates the management of any given network by providing a dynamic network topology that is displayed in a web browser. The required data from the routers is obtained with Streaming Telemetry. Streaming Telemetry is a network monitoring approach in which devices constantly send data. The routers are configured with the neighbor discovery protocol CDP. This information is required to create a network topology map that will be displayed in the frontend. The visualization of paths, a router can send data packets through, is made possible with the use of Streaming Telemetry data, which contain Segment Routing information. With Segment Routing, packets are sent from router to router through an ordered list of instructions.

The resulting web application dynamically visualizes a network topology. The nodes of this network represent routers, that upon clicking, reveal more detailed information. The application renders possible and actual segment routing paths taken between two nodes within a chosen service. It is also possible to search the history of the network. This enables the operator to see possible changes of the network at a given point in time.

Management Summary

Starting Position

For a network operator it is crucial to see the routing of data packets through a network.

Classic network monitoring tools mostly monitor the lower OSI layers with the help of SNMP, with traps and periodic polling. Some advanced network monitoring tools were even able to draw all links and devices of a network and find out which protocols were running where based on SNMP data.

With the introduction of new technologies like Segment Routing, the path taken in the network can depend on the service and its assigned requirements (e.g. QoS, bandwidth, etc.). A modern monitoring tool should be able to show these paths per service in a dynamic way.

Another technological enhancement in the network domain is Streaming Telemetry. It opens the door to near real-time monitoring of a network in a very efficient way since its using a push and not a pull model like SNMP does.

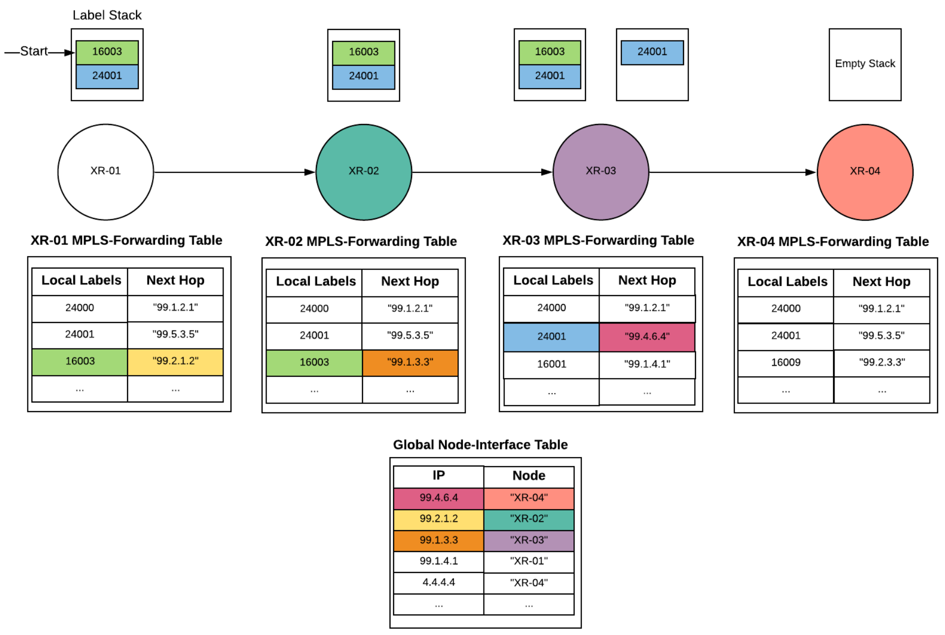
Approach, Technologies

The developed application for this bachelor thesis facilitates the management of any given network by providing a dynamic network topology that is displayed in a web browser.

The required data from the routers is obtained with Streaming Telemetry. Streaming Telemetry is a network monitoring approach in which devices constantly send data. In this case, the virtual routers send data every 5 minutes to a Cisco software called Big Muddy Pipeline.

One important information is about the neighbors of each device. The routers are configured with the neighbor discovery protocol CDP. This information is required to create a network topology map that is displayed in the frontend afterwards.

Another essential feature is the visualization of paths a router can send data packets through. This is made possible with the use of Streaming Telemetry data, which contain Segment Routing information. With Segment Routing an application directs packets through an ordered list of instructions. The ordered list is called the "Label Stack" and the instructions are "Labels". A label dictates to which router the data is sent. The sending router can look this information up in its Multiprotocol Label Switching (MPLS)-Forwarding-Table. If the destination is reached, the next label on the label stack is processed. This is repeated until the label stack is empty and the destination is reached.



XR-01 MPLS-Forwarding Table

Local Labels	Next Hop
24000	"99.1.2.1"
24001	"99.5.3.5"
16003	"99.2.1.2"
...	...

XR-02 MPLS-Forwarding Table

Local Labels	Next Hop
24000	"99.1.2.1"
24001	"99.5.3.5"
16003	"99.1.3.3"
...	...

XR-03 MPLS-Forwarding Table

Local Labels	Next Hop
24000	"99.1.2.1"
24001	"99.4.6.4"
16001	"99.1.4.1"
...	...

XR-04 MPLS-Forwarding Table

Local Labels	Next Hop
24000	"99.1.2.1"
24001	"99.5.3.5"
16009	"99.2.3.3"
...	...

Global Node-Interface Table

IP	Node
99.4.6.4	"XR-04"
99.2.1.2	"XR-02"
99.1.3.3	"XR-03"
99.1.4.1	"XR-01"
4.4.4.4	"XR-04"
...	...

Figure 1: Segment Routing

All this information is streamed from the routers through the Cisco Big Muddy Pipeline. The data output from the pipeline is stored in Apache Kafka. This data stream is then being filtered and aggregated by a stream worker to compose the needed data structure for the backend.

A Spring Boot backend consumes this processed data directly from Apache Kafka and provides an API for frontend requests.

The frontend is a Vue web application that requests data from the backend and displays the network topology with the cytoscape graph library.

All these components are containerized with Docker. This makes them easily deployable and manageable.

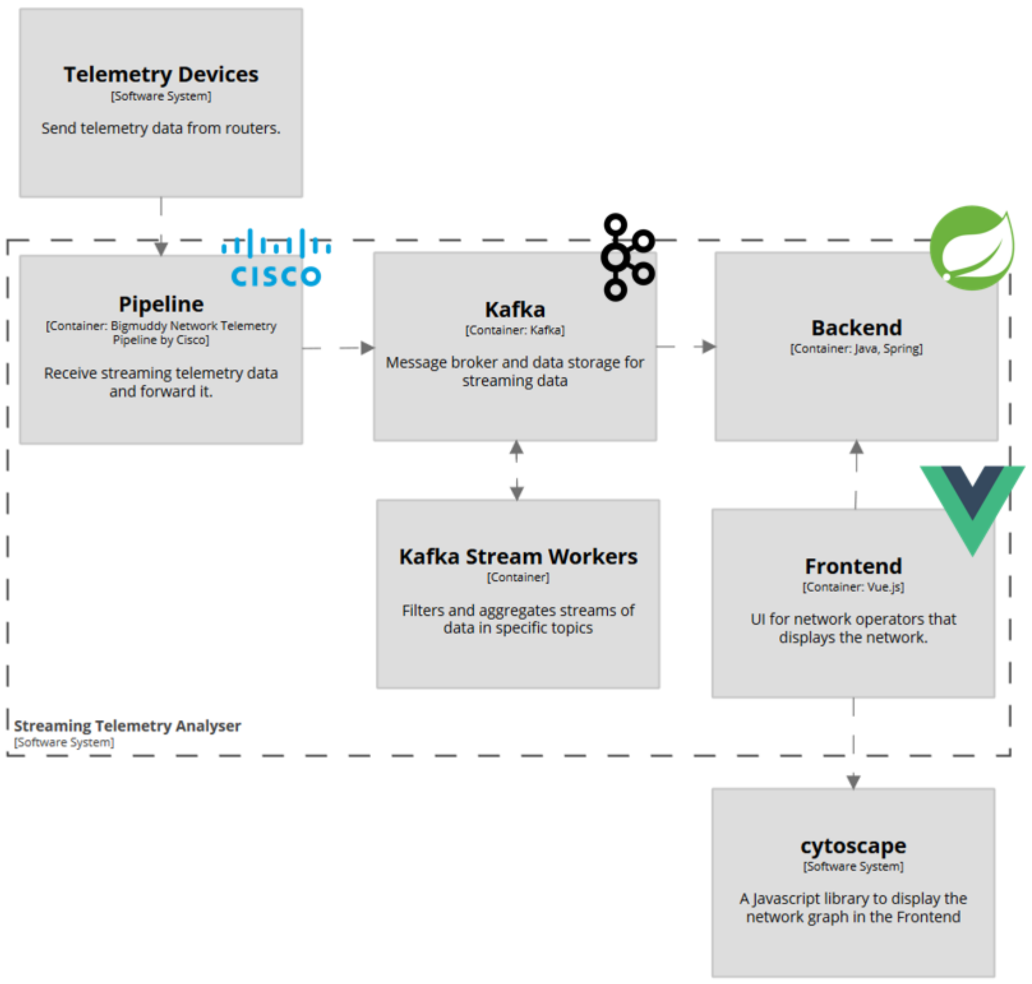

Telemetry Devices
[Software System]
Send telemetry data from routers.
cisco
Pipeline
[Container: Bigmuddy Network Telemetry Pipeline by Cisco]
Receive streaming telemetry data and forward it.
Kafka
[Container: Kafka]
Message broker and data storage for streaming data
Backend
[Container: Java, Spring]
Kafka Stream Workers
[Container]
Filters and aggregates streams of data in specific topics
Frontend
[Container: Vue.js]
UI for network operators that displays the network.
Streaming Telemetry Analyser
[Software System]
cytoscape
[Software System]
A Javascript library to display the network graph in the Frontend

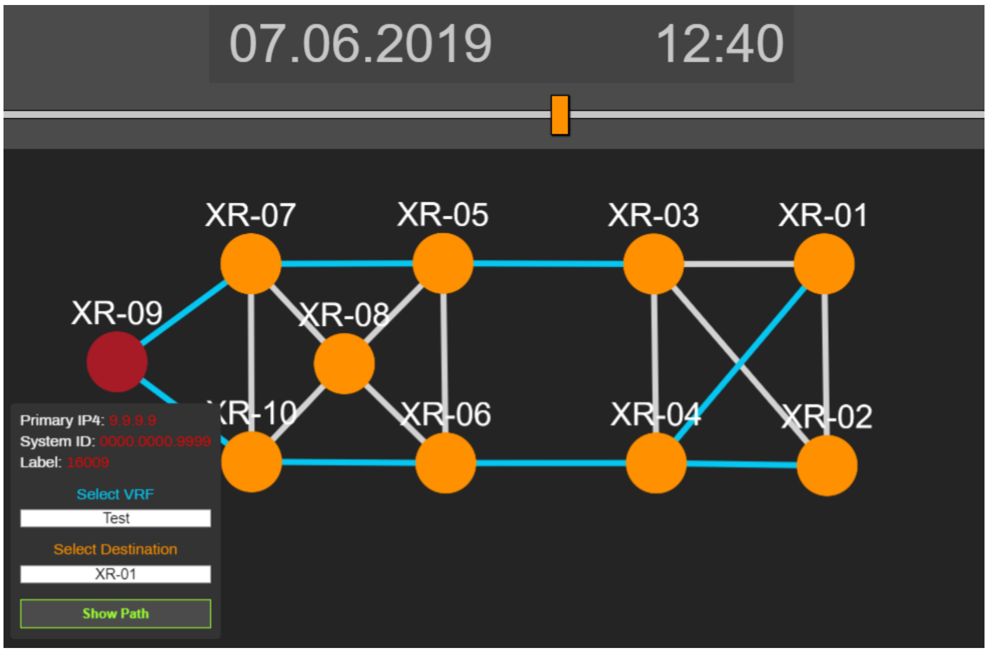
Figure 2: Architecture

Conclusion

The resulting application is based on an existing prototype which has been adapted and extended with additional features. Although the prototype presented a good base, some existing features needed to be improved and altered due to the technology and data used.

A network topology is dynamically rendered by the web application. The nodes of this network represent routers, that upon clicking, reveal more detailed information. The application visualizes possible and actual segment routing paths taken between two nodes within a chosen service.

It is also possible to search the history of the network. This enables the operator to see possible changes of the network at a given point in time. The time can be set with a date-time.picker or with a slider.



Primary IP4: 9.9.9.9
System ID: 0000.0000.9999
Label: 16009

Select VRF: Test

Select Destination: XR-01

Figure 3: Web Application

Prospect

There are a lot of possibilities for the evolution of the application with one being the testing of the application on a larger scale.

Also interesting is the filtering of events. Events could be displayed in the UI in a list as notifications with detailed information. Examples could be crashed/misbehaving routers or lost connections between routers. An approach could be to persist the topology and node information and compare it against future data. The resulting differences could then be displayed in the UI as. For future implementations it might be good to know that Streaming Telemetry has the capability to trigger events on its own.

Another big topic is the storage space. If Apache Kafka runs for a longer period of time, the storage fills up. Instead of storing all incoming data every 10 minutes, only the changes in the topology and node information are persisted. This would result in a tremendous amount of storage savings. This could be further optimized by storing data less frequently the older they get. For example within a day store data in a 10 minute interval, within a Week daily and so on.

Contents

Part I

Technical Report

Chapter 1

Introduction

This bachelor thesis is based on the thesis "Streaming Telemetry" from the HSR students Matthias Dunkel and Raffael Vögeli.

Note: Some parts in this chapter are taken from the original assignment

1.1 Problem Domain

Classic network monitoring tools mostly monitor the lower OSI layers with the help of SNMP, with traps and periodic polling. Some advanced network monitoring tools were even able to draw all links and devices of a network and find out which protocols were running where based on SNMP data.

1.2 Current Concepts

These are current concepts used in a network domain

1.2.1 Segment Routing

With the introduction of new technologies like Segment Routing, the path taken in the network can depend on the service and its assigned requirements (e.g. QoS, bandwidth, etc.). A modern monitoring tool should be able to show these paths per service in a dynamic way.

1.2.2 Streaming Telemetry

Another technological enhancement in the network domain is Streaming Telemetry. It opens the door to near real-time monitoring of a network in a very efficient way since its using a push and not a pull model like SNMP does.

The required data from the routers is obtained with Streaming Telemetry. Streaming Telemetry is a network monitoring approach in which devices constantly send data. In this case, the virtual routers send data every 5 minutes to a Cisco software called Big Muddy Pipeline.

1.2.3 Neighbour Discovery

The developed application for this bachelor thesis facilitates the management of any given network by providing a dynamic network topology that is displayed in a web browser.

One important information is about the neighbors of each device. The routers are configured with the neighbor discovery protocol CDP. This information is required to create a network topology map that is displayed in the frontend afterwards.

1.3 Vision

For a network operator it is crucial to see the routing of data packets through a network. This Application facilitates the management of such a network by providing a network topology with the corresponding information on each node and link. Furthermore it is possible to search the history of the network topology.

1.4 Goals

The goal of this bachelor thesis is to innovate the way routing is monitored. The use cases are going to be based on segment routing even if other routing technologies could be the streaming telemetry data source for future use cases.

As part of the bachelor thesis a web UI needs to be developed. The focus of this bachelor thesis is on the frontend. Nevertheless, some backend functionality will be required too. The web UI is going to display paths of services which are provided by the network.

Goal	Optional	Achieved
The web UI should be able to display the network topology. Each router must be represented with an icon and the connection between the routers with a line.	No	✓

The physical network topology should be built using CDP/LLDP Streaming Telemetry data.	No	✓
Network topology changes should be detected by the application (via the Streaming Telemetry data).	No	✓
To provide traceability of network changes a history should be maintained. The operator needs to be able to browse through the history. When the operator selects a specific time, then the data from that time must be displayed.	No	✓
The history can be searched and filtered by the operator for certain events. For example, such an event could be a path change because of a crashed router.	Yes	
The operator can select a service (VRF) on a node. The web UI then shows the operator the path of the service. If multiple service destinations are possible, all are shown by default. In addition, the operator can then select a specific destination and this way only the requested path is shown.	No	✓
An operator can select a random link and the web UI then shows him the services which are traversing that link. An additional click on a VRF of that list shows the operator the whole path of this service.	No	
It should be possible to display different detail levels of the information at different places (e.g. by hovering over a link/device: show some specific metrics).	No	✓

It should be possible for the operator to select a specific path property and the web UI then displays the best path based on the selected property. Possible properties could be lowest latency, highest bandwidth or just best effort (based on IGP metric).	Yes
It should be possible to trigger events in case of a detected network change. Such an event can then be displayed inside the web UI (e.g. in a notification area).	Yes
It should be possible to show the different FlexAlgo topologies in the web UI. Depending on a predefined FlexAlgo topology not all links or nodes are available for routing. In such a scenario operator can optimize the number of links used for a certain service (i.e. just encrypted links only)	Yes

Table 1.1: Project Goals

1.5 Scope Definition

The configuration of the test network and the analysis of the relevant sensors is not part of this thesis. This falls under the responsibility of the Institute for Networked Solutions (INS). The infrastructure and sensor information will be given to the students.

This bachelor thesis builds on the knowledge learned in a prior student research project thesis about Streaming Telemetry (took place in HS2018). The code from the student research project thesis can be used as a base for the applications which are going to be built in this bachelor thesis. Nevertheless, this is optional and can be decided by the bachelor students.

Chapter 2

Approach

This chapter elaborates on the concepts in the network domain and how they are used to accomplish the defined use cases.

2.1 YANG Models

A YANG model defines the data streamed from the routers in the network that is needed for the application. The models are divided into different encoding paths from which the data is aggregated from. The used YANG models in this thesis are in the JavaScript Object Notation (JSON) as seen for example in figure 2.1 (grey block).

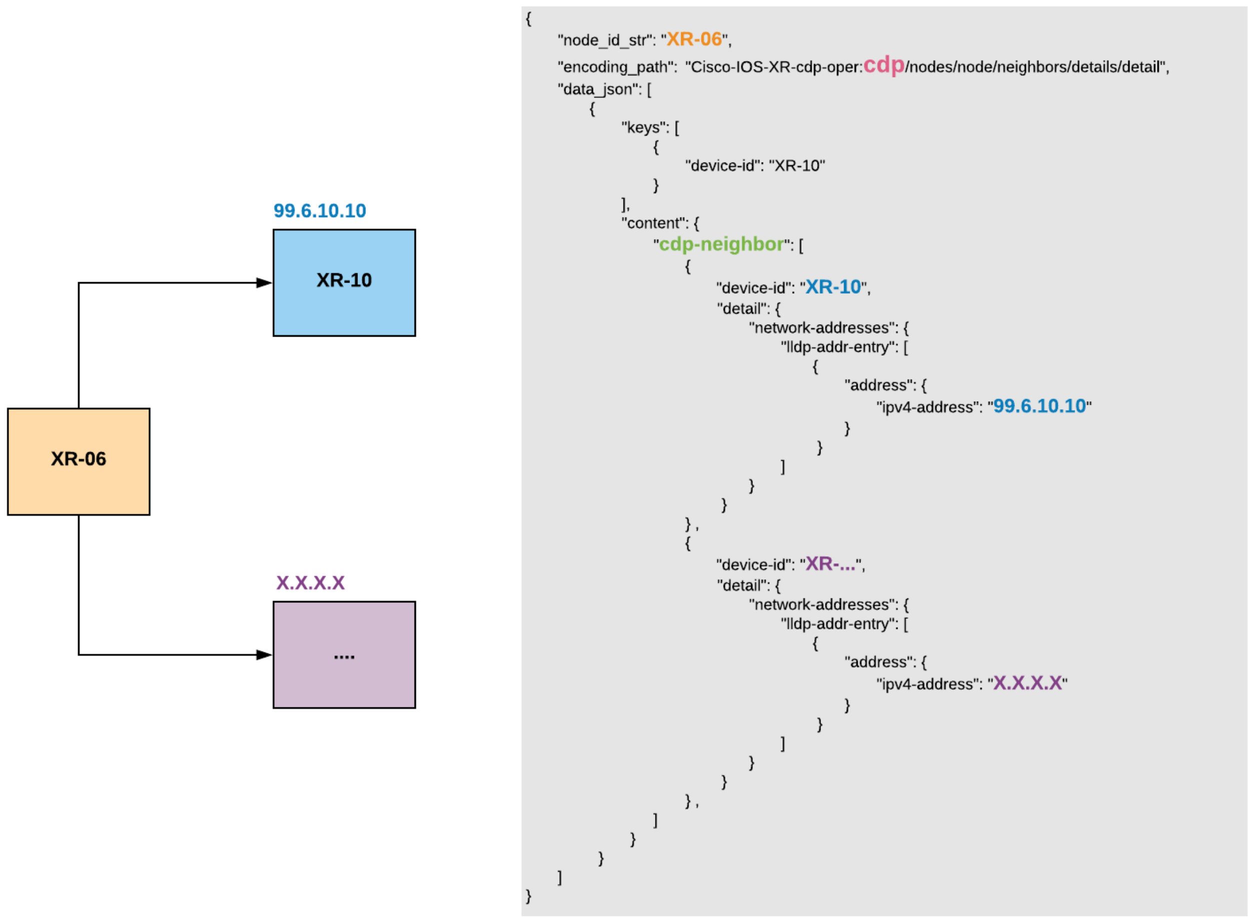
2.2 Network Topology

A network topology shows all the routers (nodes) and their connections (links) among each other. To display this network topology the Cisco Discovery Protocol is used.

2.2.1 Cisco Discovery Protocol (CDP)

CDP discovers all neighbour devices for every device in the network and the interface over which they exchange information. This information is received from the YANG model seen in figure 2.1. The information is required to create a network topology that is displayed in the frontend.

Decision: To establish the network topology the prototype used LLDP to stay independent of any company specific solution. However LLDP revealed to be unreliable on a Kernel-based-VM infrastructur when it comes to building a network, therefore the protocol has been changed to CDP.



```
{
    "node_id_str": "XR-06",
    "encoding_path":  "Cisco-IOS-XR-cdp-oper:cdp/nodes/node/neighbors/details/detail",
    "data_json": [
        {
            "keys": [
                {
                    "device-id": "XR-10"
                }
            ],
            "content": {
                "cdp-neighbor": [
                    {
                        "device-id": "XR-10",
                        "detail": {
                            "network-addresses": {
                                "lldp-addr-entry": [
                                    {
                                        "address": {
                                            "ipv4-address": "99.6.10.10"
                                        }
                                    }
                                ]
                            }
                        }
                    } ,
                    {
                        "device-id": "XR-...",
                        "detail": {
                            "network-addresses": {
                                "lldp-addr-entry": [
                                    {
                                        "address": {
                                            "ipv4-address": "X.X.X.X"
                                        }
                                    }
                                ]
                            }
                        }
                    } ,
                ]
            }
        }
    ]
}
```

Figure 2.1: CDP

2.3 Segment Routing

"The application steers its packets through an ordered list..." (Label Stack) "...of instructions..." (Labels) [1]

What follows are the most important artifacts of segment routing needed for this application.

2.3.1 Labels

Segment routing works with labels. Each label represents a segment of the segment routing path. There are two types of labels:

Global Labels (16xxx): Stay on stack for multiple hops before it is popped by the corresponding node.

Local Labels (24xxx): Popped immediately by the current node.

2.3.2 Label Stack

The label stack contains all labels needed to resolve a specific segment routing path. It is processed bottom up.

2.3.3 Next Hop IP

The next hop IP points to the next node in the segment routing path. It can be looked up in the MPLS-Forwarding Table 2.7, with the help of the current label of the label stack. The IP can be mapped with the help of the Node-Interface Table 2.8.

2.3.4 Multiprotocol Label Switching-Forwarding (MPLS) Table

MPLS maps the labels of a node with the corresponding Next Hop. When the node receives a label, it can look up the next hop in the table.

The MPLS-Forwarding table stores all nodes with all their available labels mapped to the "Next Hop" IP. (See 2.7)

It is needed to get the segment routing path over a specific tunnel interface with the help of the label stack.

2.3.5 Node-Interface Table

The Node-Interface table stores all nodes with all their available interfaces. (See 2.8) It is needed to look up the primary IP of the "Next Hop" IP to take the next step of the path.

2.3.6 Path Finding

In order to resolve a segment routing path in a specific VRF between two routers, the label stack and next hop IP are needed.

To get this information, follow the steps in figure 2.2 below:

1. Get Loopback IP Policy Label (YANG excerpt 2.3)
2. Get Tunnel Interface (YANG excerpt 2.4)
3. Get Label Stack and Next Hop IP (YANG excerpt 2.7)

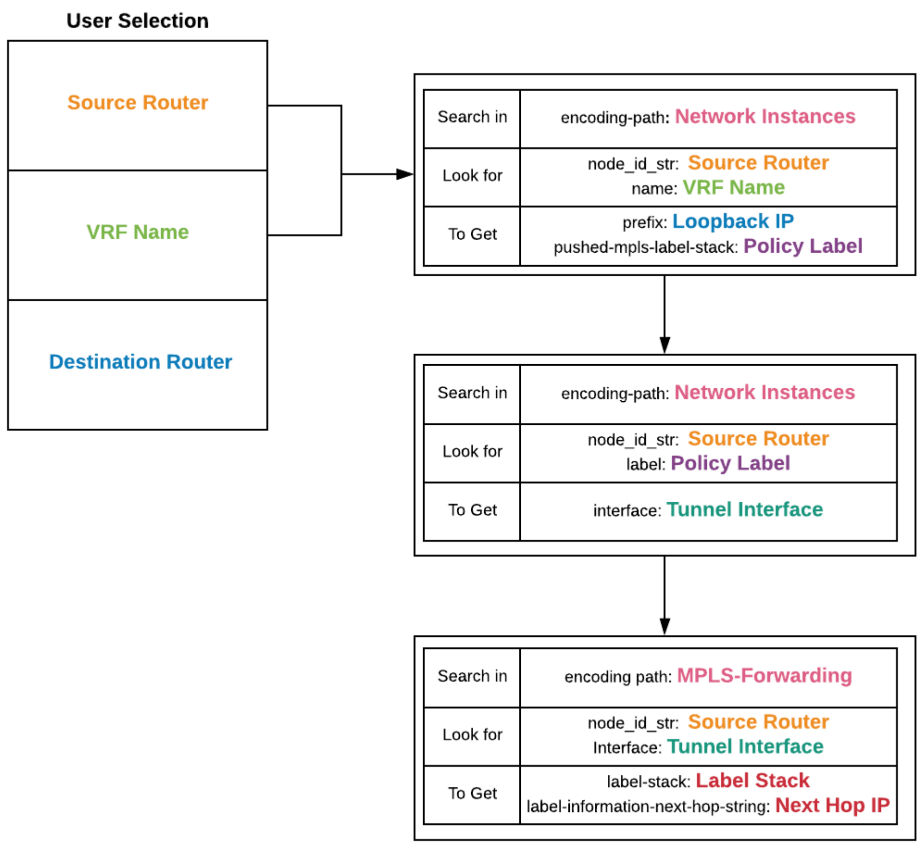
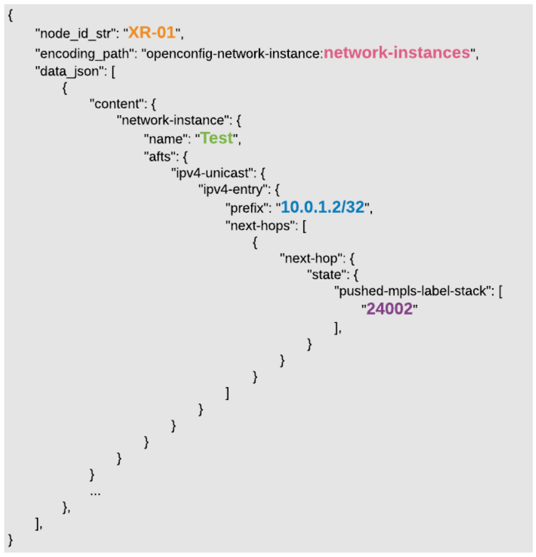
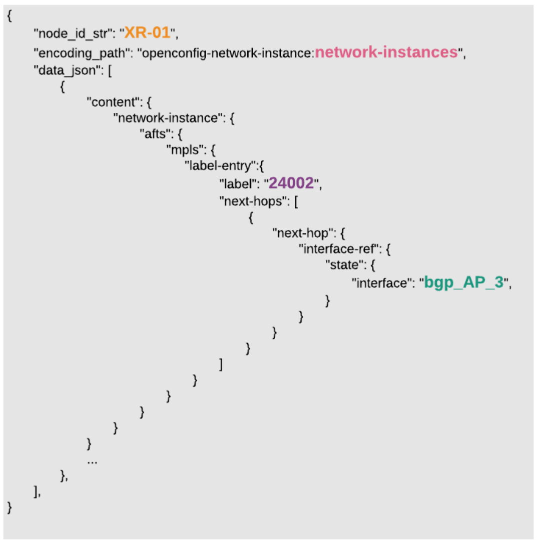

User Selection
Source Router
VRF Name
Destination Router
Search in
encoding-path: Network Instances
Look for
node_id_str: Source Router
name: VRF Name
To Get
prefix: Loopback IP
pushed-mpls-label-stack: Policy Label
Search in
encoding-path: Network Instances
Look for
node_id_str: Source Router
label: Policy Label
To Get
interface: Tunnel Interface
Search in
encoding path: MPLS-Forwarding
Look for
node_id_str: Source Router
Interface: Tunnel Interface
To Get
label-stack: Label Stack
label-information-next-hop-string: Next Hop IP

Figure 2.2: Label Stack and Next Hop



```
{
    "node_id_str": "XR-01",
    "encoding_path": "openconfig-network-instance:network-instances",
    "data_json": [
        {
            "content": {
                "network-instance": {
                    "name": "Test",
                    "afts": {
                        "ipv4-unicast": {
                            "ipv4-entry": {
                                "prefix": "10.0.1.2/32",
                                "next-hops": [
                                    {
                                        "next-hop": {
                                            "state": {
                                                "pushed-mpls-label-stack": [
                                                    "24002"
                                                ],
                                            }
                                        }
                                    }
                                ]
                            }
                        }
                    }
                }
            }
            ...
        },
    ],
}
```

Figure 2.3: Policy Label YANG



```
{
    "node_id_str": "XR-01",
    "encoding_path": "openconfig-network-instance:network-instances",
    "data_json": [
        {
            "content": {
                "network-instance": {
                    "afts": {
                        "mpls": {
                            "label-entry":{
                                "label": "24002",
                                "next-hops": [
                                    {
                                        "next-hop": {
                                            "interface-ref": {
                                                "state": {
                                                    "interface": "bgp_AP_3",
                                                }
                                            }
                                        }
                                    }
                                ]
                            }
                        }
                    }
                }
            }
            ...
        },
    ],
}
```

Figure 2.4: Tunnel Interface YANG

Now that the label stack and next hop IP have been acquired, the path can be resolved as shown in Figure 2.5.

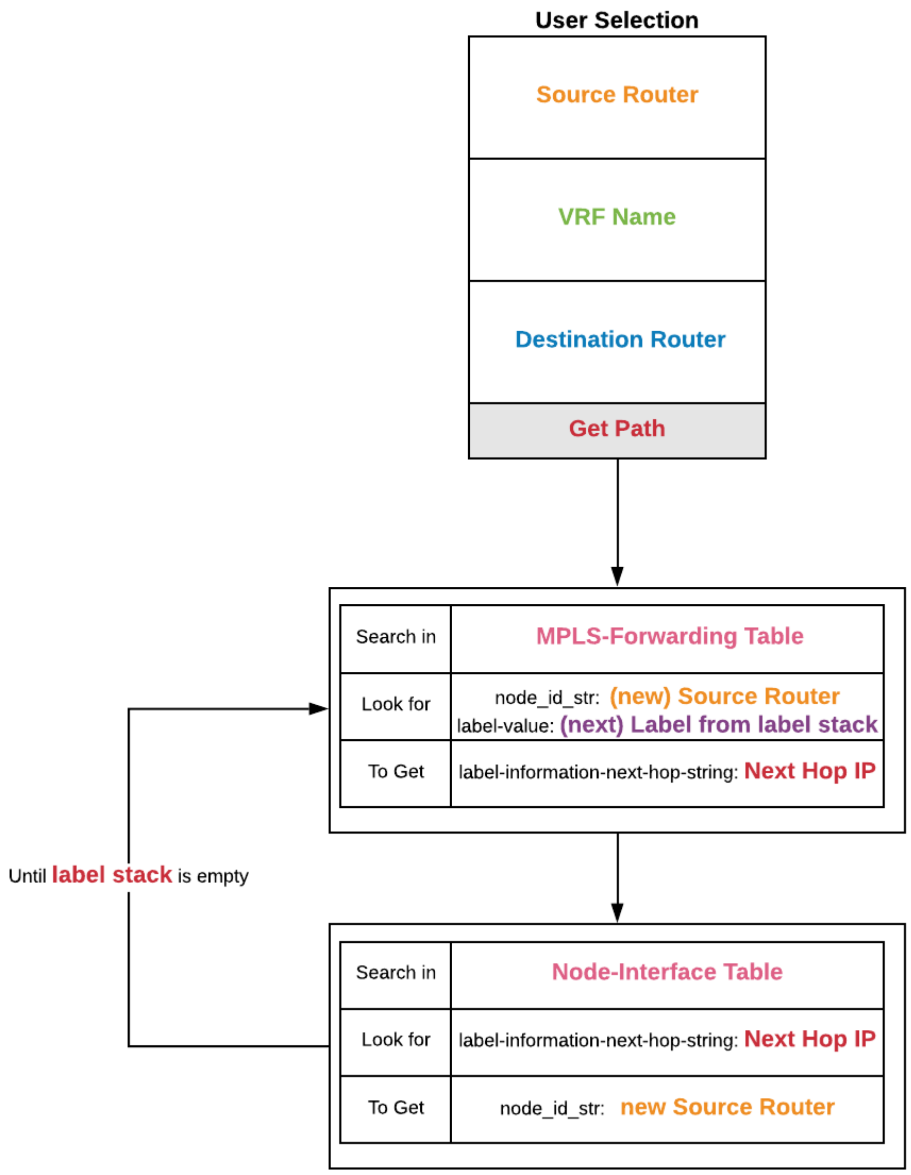

User Selection
Source Router
VRF Name
Destination Router
Get Path
Search in
MPLS-Forwarding Table
Look for
node_id_str: (new) Source Router
label-value: (next) Label from label stack
To Get
label-information-next-hop-string: Next Hop IP
Until label stack is empty
Search in
Node-Interface Table
Look for
label-information-next-hop-string: Next Hop IP
To Get
node_id_str: new Source Router

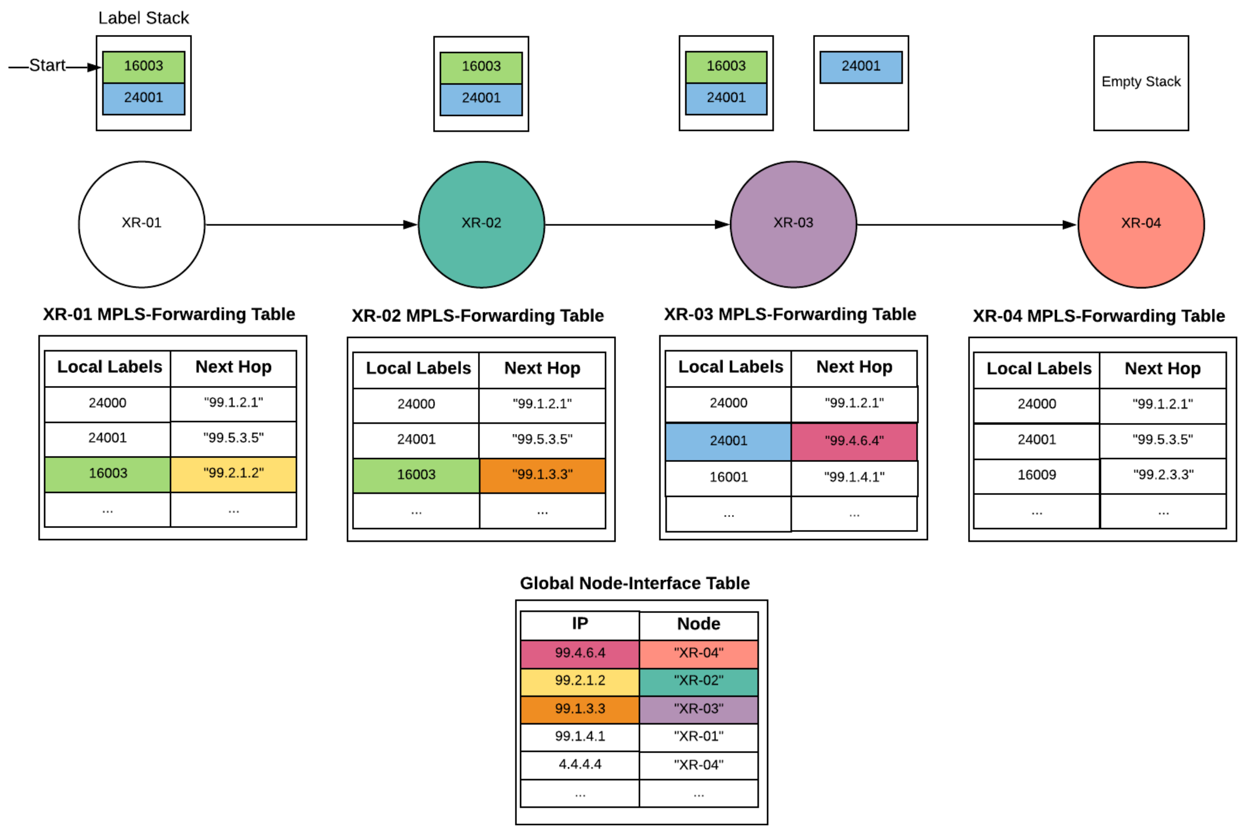
Figure 2.5: Get Path

The Label Stack is processed bottom up. The first (lowermost) label is looked up in the MPLS-forwarding table (See 2.7) to find the corresponding "Next

Hop" IP-address. This IP-address needs to be mapped in the node-interface Table (See 2.8).

Depending on the label type, the label is popped (local label) or it stays on top of the stack (global label). The label stack is forwarded to the next hop, until the entire stack has been processed.

Figure 2.6 gives an overview of the process for a given source and destination router over a specified VRF.



XR-01 MPLS-Forwarding Table

Local Labels	Next Hop
24000	"99.1.2.1"
24001	"99.5.3.5"
16003	"99.2.1.2"
...	...

XR-02 MPLS-Forwarding Table

Local Labels	Next Hop
24000	"99.1.2.1"
24001	"99.5.3.5"
16003	"99.1.3.3"
...	...

XR-03 MPLS-Forwarding Table

Local Labels	Next Hop
24000	"99.1.2.1"
24001	"99.4.6.4"
16001	"99.1.4.1"
...	...

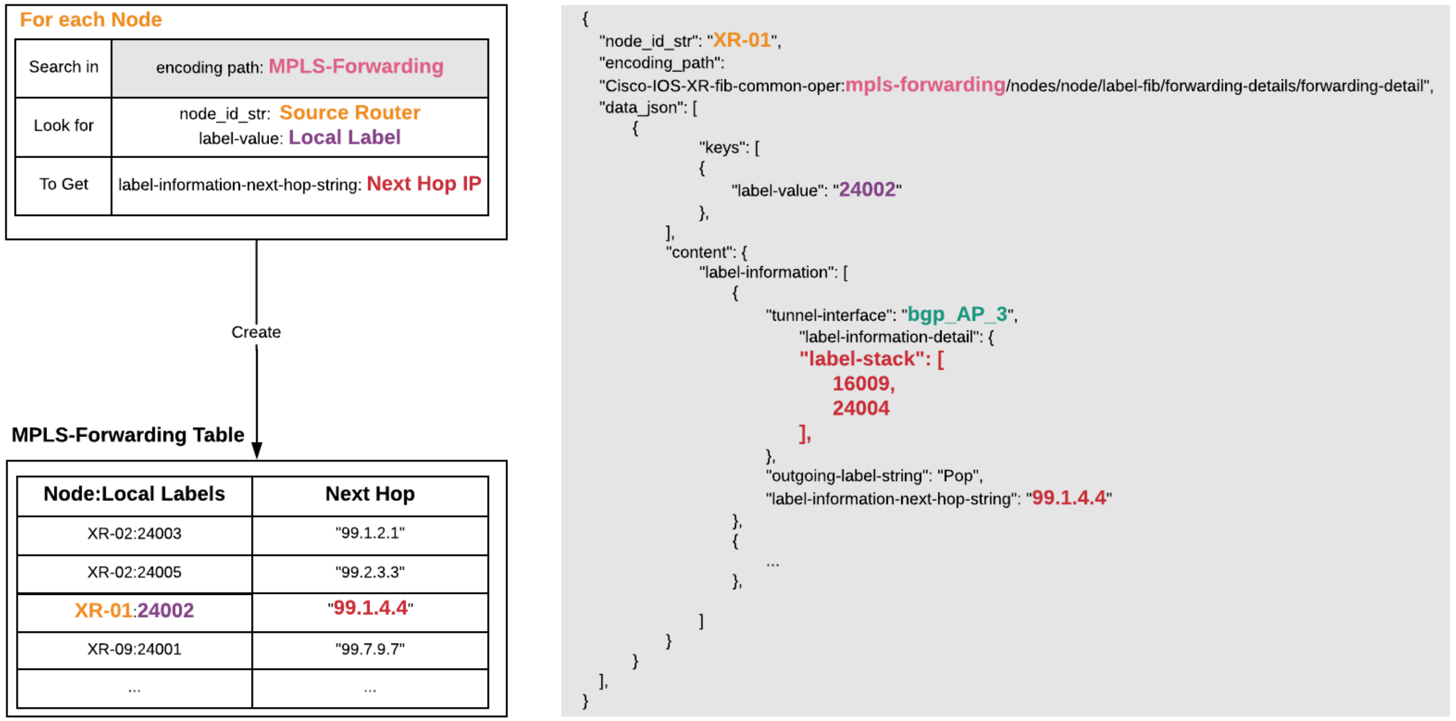
XR-04 MPLS-Forwarding Table

Local Labels	Next Hop
24000	"99.1.2.1"
24001	"99.5.3.5"
16009	"99.2.3.3"
...	...

Global Node-Interface Table

IP	Node
99.4.6.4	"XR-04"
99.2.1.2	"XR-02"
99.1.3.3	"XR-03"
99.1.4.1	"XR-01"
4.4.4.4	"XR-04"
...	...

Figure 2.6: Overview



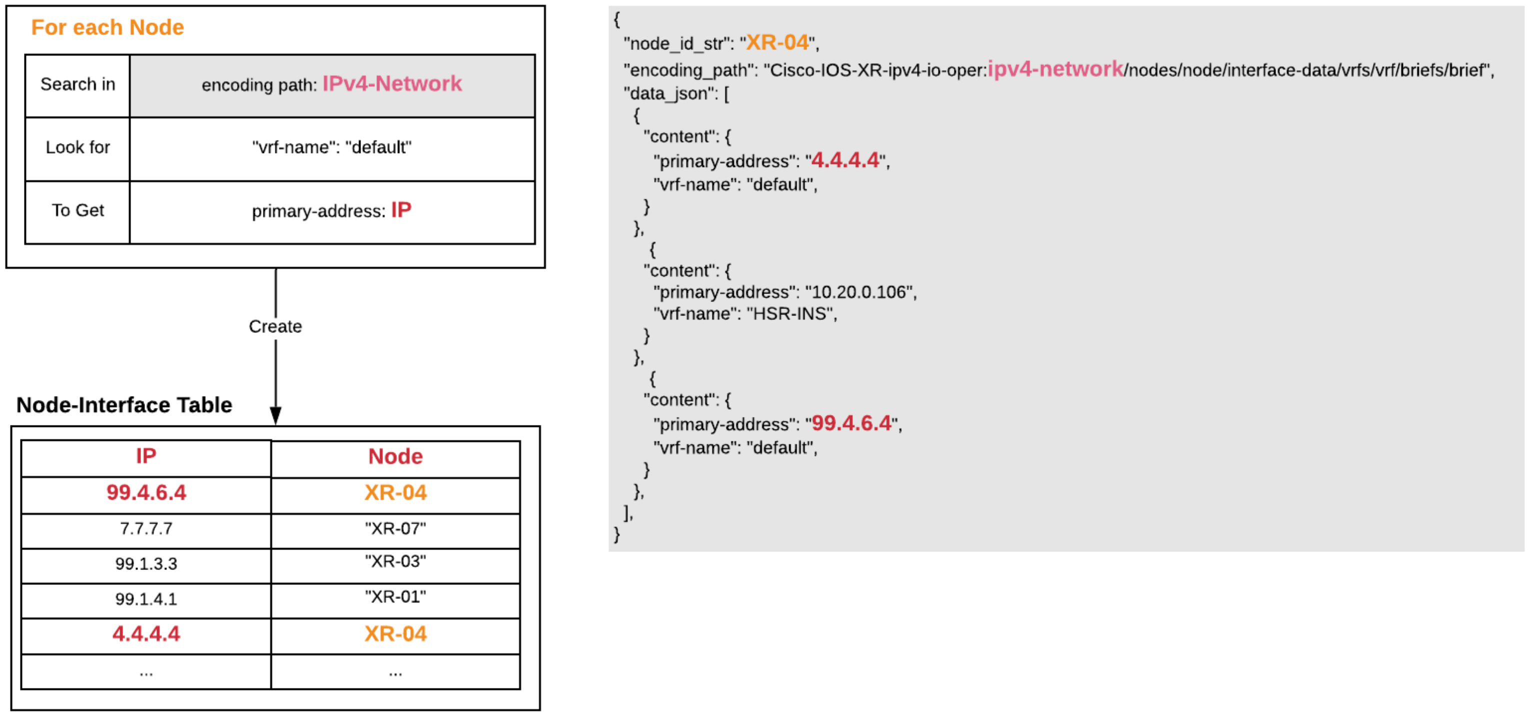
Search in	encoding path: MPLS-Forwarding
Look for	node_id_str: Source Router label-value: Local Label
To Get	label-information-next-hop-string: Next Hop IP

Node:Local Labels	Next Hop
XR-02:24003	"99.1.2.1"
XR-02:24005	"99.2.3.3"
XR-01:24002	"99.1.4.4"
XR-09:24001	"99.7.9.7"
...	...

```
{
  "node_id_str": "XR-01",
  "encoding_path":
  "Cisco-IOS-XR-fib-common-oper:mpls-forwarding/nodes/node/label-fib/forwarding-details/forwarding-detail",
  "data_json": [
      {
              "keys": [
              {
                  "label-value": "24002"
              },
          ],
          "content": {
              "label-information": [
                  {
                      "tunnel-interface": "bgp_AP_3",
                          "label-information-detail": {
                          "label-stack": [
                              16009,
                              24004
                          ],
                      },
                      "outgoing-label-string": "Pop",
                      "label-information-next-hop-string": "99.1.4.4"
                  },
                  {
                      ...
                  },

              ]
          }
      }
  ],
}
```

Figure 2.7: MPLS Table



Search in	encoding path: IPv4-Network
Look for	"vrf-name": "default"
To Get	primary-address: IP

Node-Interface Table

IP	Node
99.4.6.4	XR-04
7.7.7.7	"XR-07"
99.1.3.3	"XR-03"
99.1.4.1	"XR-01"
4.4.4.4	XR-04
...	...

```
{
  "node_id_str": "XR-04",
  "encoding_path": "Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node/interface-data/vrfs/vrf/briefs/brief",
  "data_json": [
   {
    "content": {
      "primary-address": "4.4.4.4",
      "vrf-name": "default",
    }
   },
     {
    "content": {
      "primary-address": "10.20.0.106",
      "vrf-name": "HSR-INS",
    }
   },
     {
    "content": {
      "primary-address": "99.4.6.4",
      "vrf-name": "default",
    }
   },
  ],
}
```

Figure 2.8: Node-Interface Table

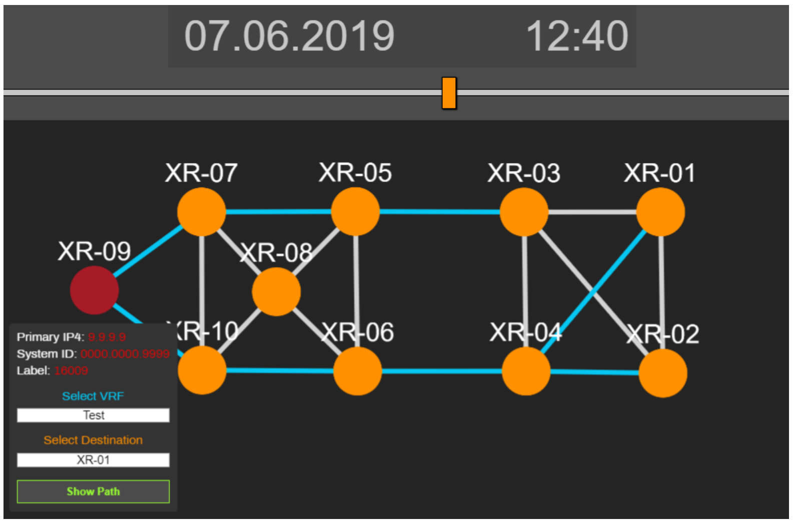
Chapter 3

Conclusion

The resulting application is based on an existing prototype which has been adapted and extended with additional features. Although the prototype presented a good base, some existing features needed to be improved and altered due to the technology and data used.

A network topology is dynamically rendered by the web application. The nodes of this network represent routers, that upon clicking, reveal more detailed information. The application visualizes possible and actual segment routing paths taken between two nodes within a chosen service.

It is also possible to search the history of the network. This enables the operator to see possible changes of the network at a given point in time. The time can be set with a date-time.picker or with a slider.



Primary IP4: 9.9.9.9
System ID: 0000.0000.9999
Label: 16009
Select VRF: Test
Select Destination: XR-01

Figure 3.1: Web Application

Chapter 4

Prospect

Apart from the possible features listed below, the application should be taken to test on a bigger scale to see how scalable the solution is.

4.1 Event Filtering

Events could be displayed in the UI in a list as notifications with information. Examples could be crashed/misbehaving routers or lost connections between routers.

To achieve this the topology and node information could be persisted and compared against future data. The resulting differences could then be send displayed in the UI as notifications.

Streaming Telemetry has the capability to trigger events on its own. This could be used for future implementations.

4.2 Save Storage

Another big topic is the storage space. If Apache Kafka runs for a longer period, the storage fills up. Instead of storing all incoming data every 10 minutes, only the changes in the topology and node information are persisted. This would result in a tremendous amount of storage savings.

One approach could be to store the incoming streams into KTables. A KTable is a Kafka object that stores key-value pairs and updates the value of a certain key if it does not match the stored value instead of concatenating the new entries like in a KStream. This results in a reduction of the accumulated data. This could be further optimized by storing data less frequently the older they are. For example within a day store data in a 10 minute interval, within a Week daily and so on.

4.3 Path Properties

The user should be able to select a specific property for a service path between two nodes. The properties could be lowest latency, highest bandwidth or best effort.

Part II

Project Documentation

Chapter 5

Analysis

5.1 Requirement Analysis

The following Use Cases (table 5.1) and Non-Functional Requirements (table 5.2) were defined by the advisor and external partners.

5.1.1 Use Cases

ID	Use Case	Opt.	Prio.	Done
UC-1	The web UI should be able to display the network topology. Each router must be represented with an icon and the connection between the routers with a line.	No	3*	✓
UC-2	The physical network topology should be built using CDP/LLDP Streaming Telemetry data.	No	3*	✓
UC-3	Network topology changes should be detected by the application (via the Streaming Telemetry data).	No	3*	✓
UC-4	To provide traceability of network changes a history should be maintained. The operator needs to be able to browse through the history. When the operator selects a specific time, then the data from that time must be displayed.	No	2*	✓

UC-5	The history can be searched and filtered by the operator for certain events. For example, such an event could be a path change because of a crashed router.	Yes	3	
UC-6	The operator can select a service (VRF) on a node. The web UI then shows the operator the path of the service. If multiple service destinations are possible, all are shown by default. In addition, the operator can then select a specific destination and this way only the requested path is shown.	No	3	✓
UC-7	An operator can select a random link and the web UI then shows him the services which are traversing that link. An additional click on a VRF of that list shows the operator the whole path of this service.	No	2	(See 5.1.1)
UC-8	It should be possible to display different detail levels of the information at different places (e.g. by hovering over a link/device: show some specific metrics).	No	2	✓
UC-9	It should be possible for the operator to select a specific path property and the web UI then displays the best path based on the selected property. Possible properties could be lowest latency, highest bandwidth or just best effort (based on IGP metric).	Yes	3	
UC-10	It should be possible to trigger events in case of a detected network change. Such an event can then be displayed inside the web UI (e.g. in a notification area).	Yes	2	

UC-11	It should be possible to show the different FlexAlgo topologies in the web UI. Depending on a predefined FlexAlgo topology not all links or nodes are available for routing. In such a scenario operator can optimize the number of links used for a certain service (i.e. just encrypted links only)	Yes	1

Table 5.1: Use Cases

* These use cases are dependencies for some other use cases. Therefore, they are rated as high priority even though they do not require much (engineering) work to do if the d3.js graph framework code from the student research project thesis is taken. This means that this use cases will have more or less weight for the rating depending on the invested work.

UC-7

To realise UC-7 it is mandatory to know which services travel through each link, while processing the telemetry topics. Otherwise it would be necessary to calculate all paths between all nodes for each service. This way of finding all services on a link does not perform well and is by no way scalable. Unfortunately, this consideration was not made while designing and implementing the first Use-Cases.

In the end, the redesign of the topic aggregation took to much time and aside from the Node topic was not implemented. The redesigned concept contains only three topics which have all the needed information by themselves. The largest improvement brings the persistent of the complete routes between nodes as topic records. With this it is easy to find all services which travel through a link.

Here is a draft of the intended changes of the topics:

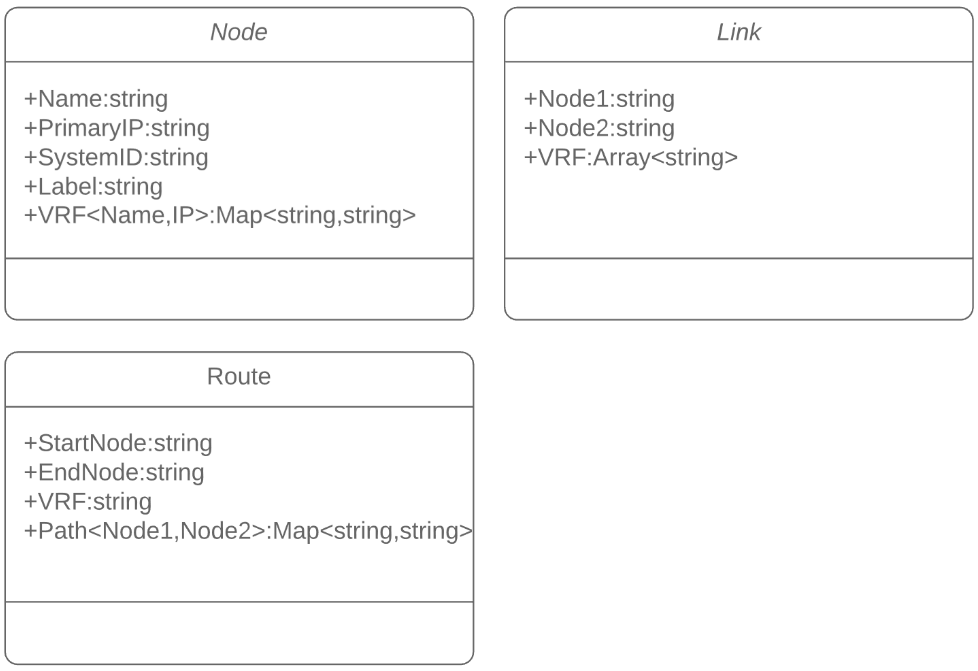


Figure 5.1: Redesigned Topics Domain Model

5.1.2 Use Case Diagram

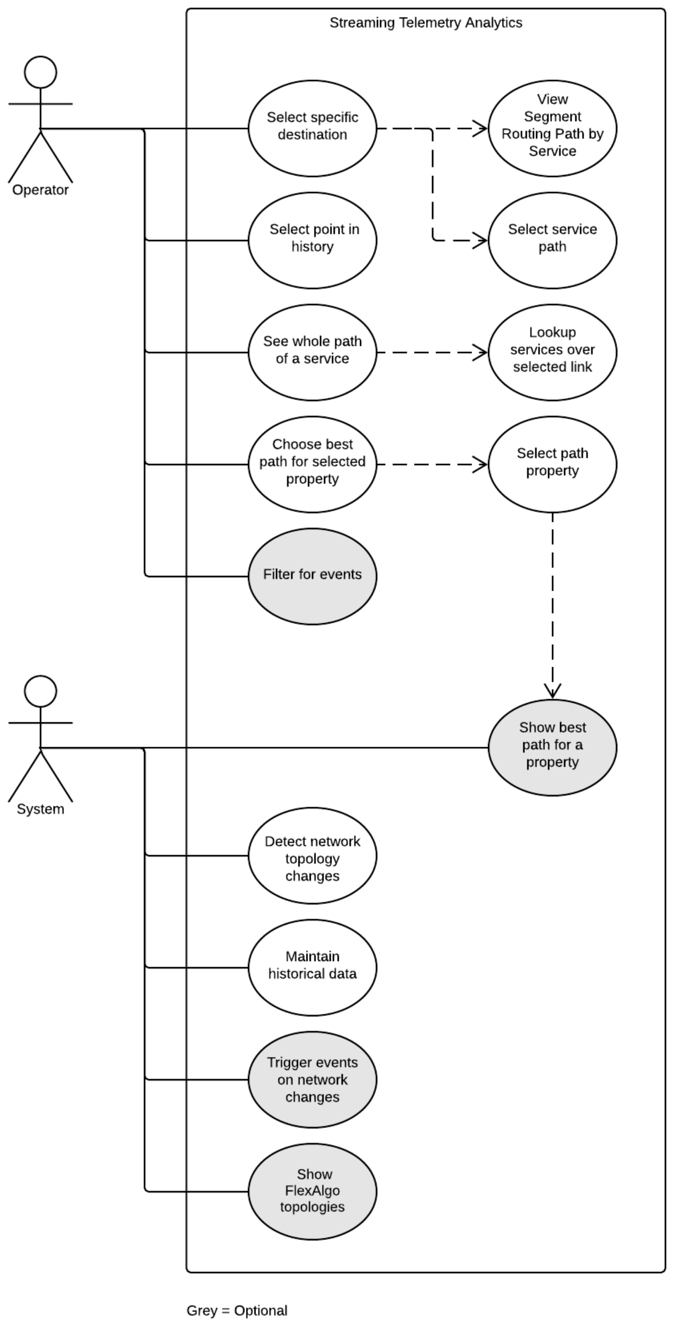

Streaming Telemetry Analytics
Operator
Select specific destination
View Segment Routing Path by Service
Select point in history
Select service path
See whole path of a service
Lookup services over selected link
Choose best path for selected property
Select path property
Filter for events
System
Show best path for a property
Detect network topology changes
Maintain historical data
Trigger events on network changes
Show FlexAlgo topologies
Grey = Optional

Figure 5.2: Use Case Diagram

5.1.3 Non-Functional Requirements

ID	NFR
NFR-1	The solution architecture should scale well (100 routers and more).
NFR-2	For a simplified deployment the applications must be runnable as dockerized microservices.

Table 5.2: Non-Functional Requirements

NFR Testing

The NFR testing is rather cumbersome because there is a fixed number of routers (10) configured from the advisor. That makes it not possible for NFR-1 to be tested.

For NFR-2 on the other hand, it is possible to log on to the server and see if the docker containers are up and running with "docker ps -a".

5.2 Domain Analysis

See Chapter 2 Approach

5.3 Frontend Analysis

5.3.1 Framework

The Vue.js framework will be used due to the fact that the existing prototype is build with it and that this lightweight framework suffices for the scope of this project.

The React framework would have been a viable alternative for the reason that the development team already worked with React on prior projects.

Angular is to big of a framework for the scope of this project and was not considered.

5.3.2 Graph Library

The cytoscape library has been selected because a Vue.js plugin is available, it is well documented and maintained and has multiple extensions available that serves the purpose of this project.

Criteria

The following features are necessary to achieve the given use cases and therefore a must-have for the chosen graph library:

- Display network topology.
- Routers are represented by an icon.
- Show selection of service when node is clicked.
- Show/hide specific path.
- Show a list on a link selection.
- Highlight a path.

Decision

With the help of a utility value analysis (see Table 5.3), and an analysis of our Co-Advisor Philip Schmid conducted on a previous, similar problem domain, the d3 library seemed most suitable for this project.

After a first attempt to implement even the smallest of features, the d3-library revealed to be rather cumbersome for more complex implementations. This led to the decision to change the library for the runner-up library cytoscape.

Cytoscape is more applicable to the Vue.js framework because there is a component based plugin available specifically for Vue.js.

This decision resulted in a loss of one workday.

Utility Value Analysis

		visjs		cytoscape		Cisco neXt		d3	
Criterion	Weight	Grade	Points	Grade	Points	Grade	Points	Grade	Points
Community	0.1	3	0.3	4	0.4	2	0.2	6	0.6
Documentation	0.2	5	1	5	1	5	1	5	1
Features/Extensions	0.25	4	1	5	1.25	4	1	5	1.25
Framework-Adaptability	0.15	5	0.75	6	0.9	3	0.45	5	0.75
Look and Feel	0.2	4	0.8	5	1.2	5	1	4	0.8
Prototype Available	0.1	1	0.1	1	0.1	1	0.1	6	0.6
			3.95		4.85		3.75		5

Table 5.3: Graph Library Utility Value Analysis

Chapter 6

Architecture

6.1 Context

Routers are sending telemetry data to the Stream Telemetry Analyser where the data is processed and displayed in a web application.

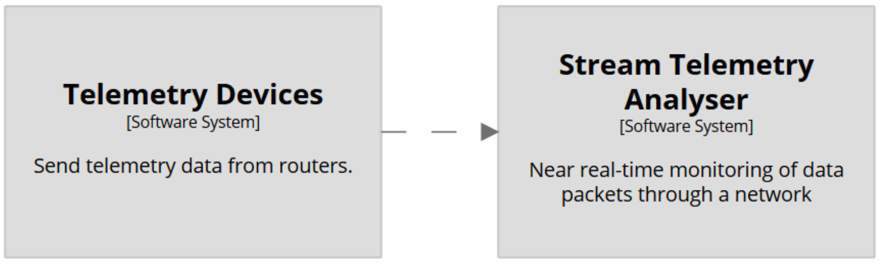

Telemetry Devices
[Software System]
Send telemetry data from routers.
Stream Telemetry Analyser
[Software System]
Near real-time monitoring of data packets through a network

Figure 6.1: Context diagram

6.2 Container

The Stream Telemetry Analyser is composed of five components. All of them are containerized with Docker. This allows the components to be easaliy deployed.

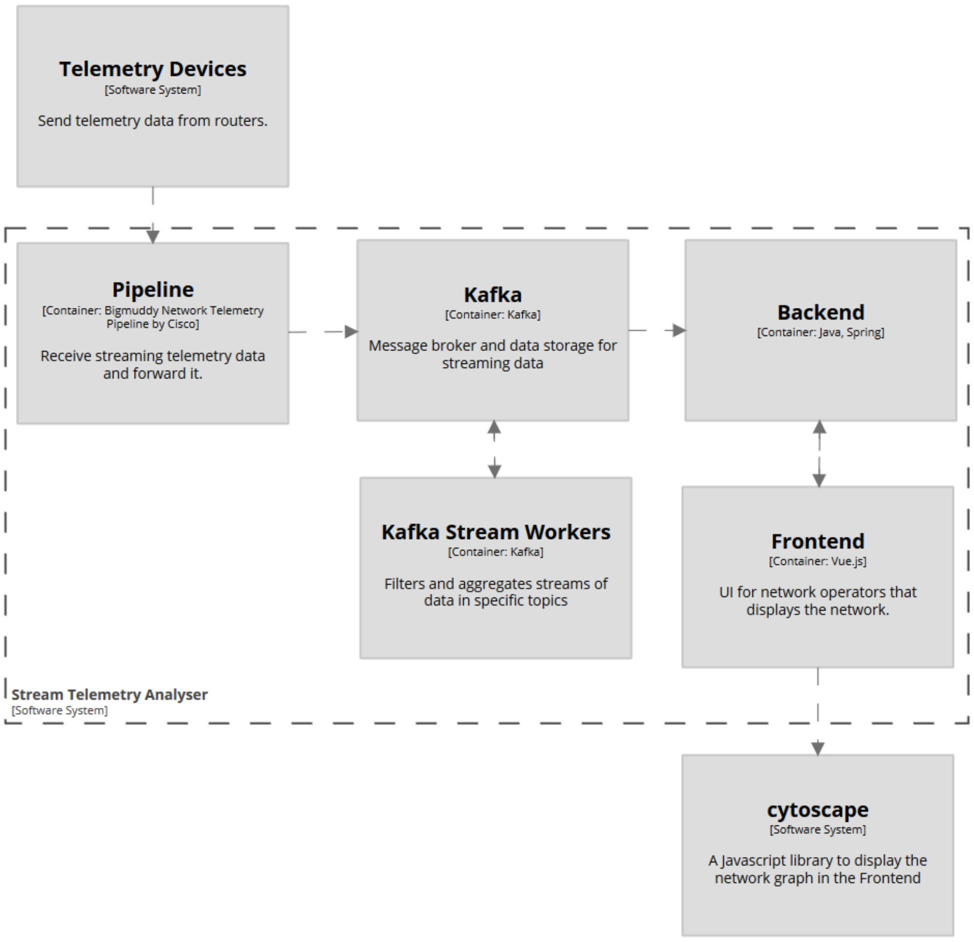

Telemetry Devices
[Software System]
Send telemetry data from routers.
Pipeline
[Container: Bigmuddy Network Telemetry Pipeline by Cisco]
Receive streaming telemetry data and forward it.
Kafka
[Container: Kafka]
Message broker and data storage for streaming data
Backend
[Container: Java, Spring]
Kafka Stream Workers
[Container: Kafka]
Filters and aggregates streams of data in specific topics
Frontend
[Container: Vue.js]
UI for network operators that displays the network.
Stream Telemetry Analyser
[Software System]
cytoscape
[Software System]
A Javascript library to display the network graph in the Frontend

Figure 6.2: Container diagram

6.3 Components

6.3.1 Telemetry Devices

The telemetry devices are routers that build a virtual network.

The routers sends data via the sensor paths. These are the paths needed:

- Cisco-IOS-XR-cdp-oper:cdp/nodes/node/neighbors
  /summaries/summary
  (For Topology)
- Cisco-IOS-XR-ipv4-io-oper:ipv4-network/nodes/node
  /interface-data/vrfs/vrf/briefs/brief
  (For Node/IP Mapping)
- openconfig-network-instance:network-instances
  (For Policy Label and Loopback IP)
- Cisco-IOS-XR-ipv4-bgp-oper:bgp/instances/instance/instance-active
  /vrfs/vrf/afs/af/path-table/path
  (For BGP Label)
- Cisco-IOS-XR-fib-common-oper:mpls-forwarding/nodes/node
  /label-fib/forwarding-details/forwarding-detail
  (For Label Stack and Next-Hop-IP)

6.3.2 Pipeline

The Cisco Bigmuddy Telemetry Pipeline is a black box piece of software. It receives the telemetry data and forwards it to Kafka.

6.3.3 Kafka

Kafka receives the raw data via the "telemetry" Topic from the pipeline. A Kafka Topic is a container where streams can publish records into and subscribers can consume these topics.

6.3.4 Kafka Stream Workers

The Kafka Stream Workers filter the raw data from Kafka in to different topics and enriches it for the backend to use. (See Kafka component diagram 6.6)

The following topics are produced from the Kafka Stream Workers:

- telemetry-mpls-table
- telemetry-node-interface-table
- telemetry-cdp-grouped

- telemetry-nodes
- telemetry-sr-path

6.3.5 Backend

The backend subscribes to the topics from the Kafka Stream Workers. These Topics are then converted into windowed topics so they can be filtered by a 10 minute window.

The Backend prepares them for HTTP requests (See 6.5) from the frontend and returns the streaming data. (See backend component diagram 6.7)

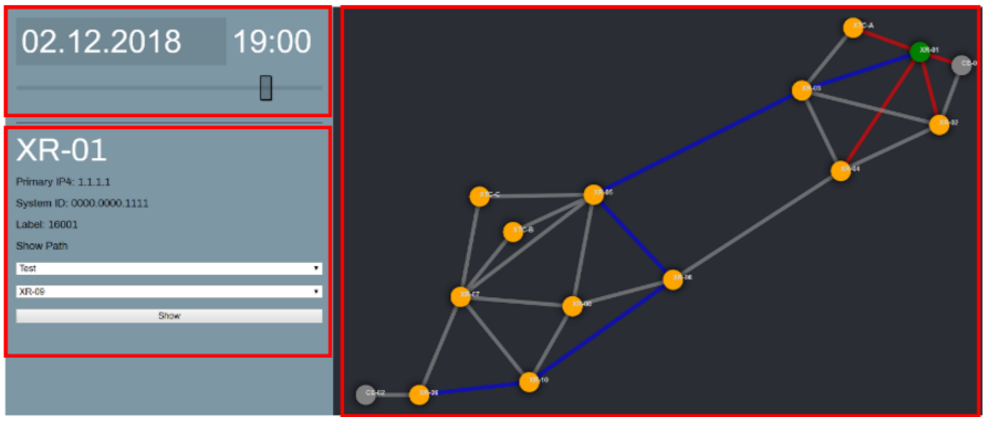
6.3.6 Frontend

The frontend sends HTTP requests to the backend and displays the UI for the client. (See frontend component diagram 6.8)

6.3.7 Layout

Old UI

The prototype UI was composed of three components: A Time-Slider, the node details and the topology graph. (See red rectangles in figure 6.3)



02.12.2018 19:00

XR-01

Primary IP4: 1.1.1.1

System ID: 0000.0000.1111

Label: 16001

Show Path

Test

XR-09

Figure 6.3: SA UI Layout

Decision

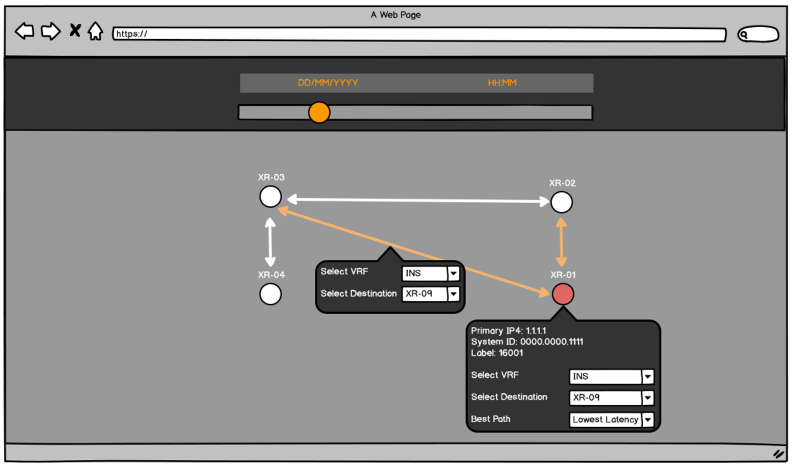
In order to increase the usability a decision was made to move the node-details-component directly into a tool-tip of the corresponding node.

The tool-tip is an extension for the cytoscape library, namely Popper.js and Tippy.js.

New UI

The new UI moves the details-component into a tool-tip. This results in a decrease of mouse movement and gives the topology more space to be worked on.

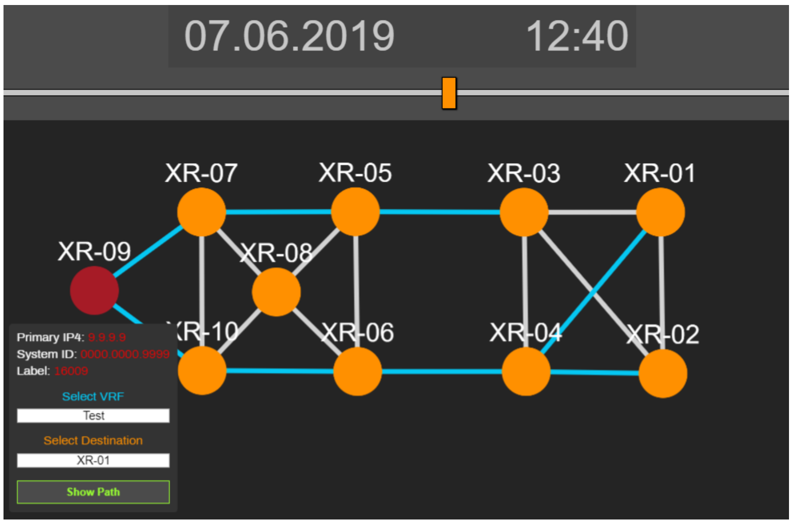
The UI uses high contrast colors for clear visibility and considers users with color vision deficiency so that important colors are distinguishable for them.



Select VRF: INS
Select Destination: XR-09

Primary IP4: 1.1.1.1
System ID: 0000.0000.1111
Label: 16001
Select VRF: INS
Select Destination: XR-09
Best Path: Lowest Latency

Figure 6.4: BA UI Mockup



Primary IP4: 9.9.9.9
System ID: 0000.0000.9999
Label: 16009
Select VRF
Test
Select Destination
XR-01

Figure 6.5: Web Application

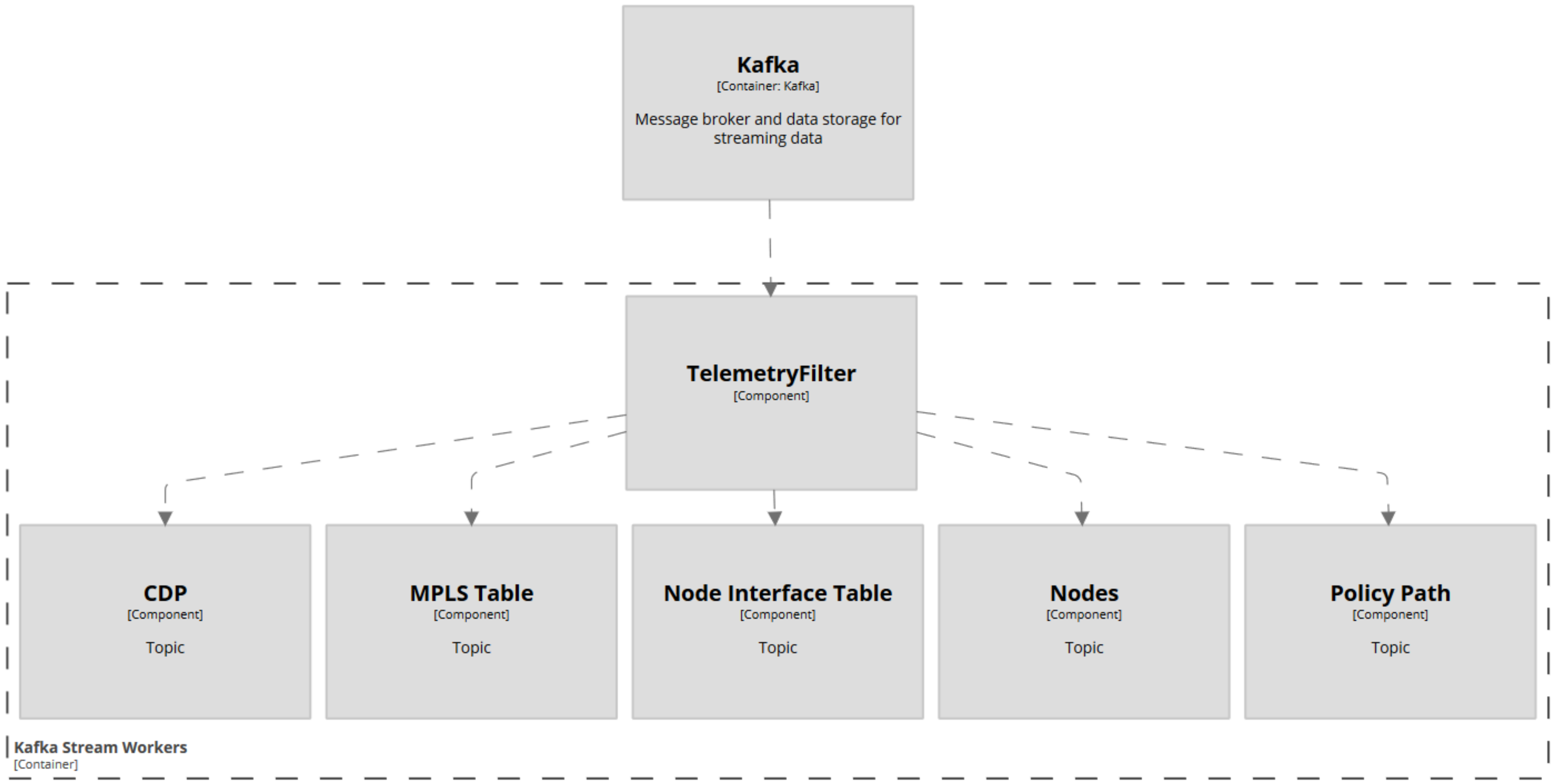


Figure 6.6: Kafka Component diagram

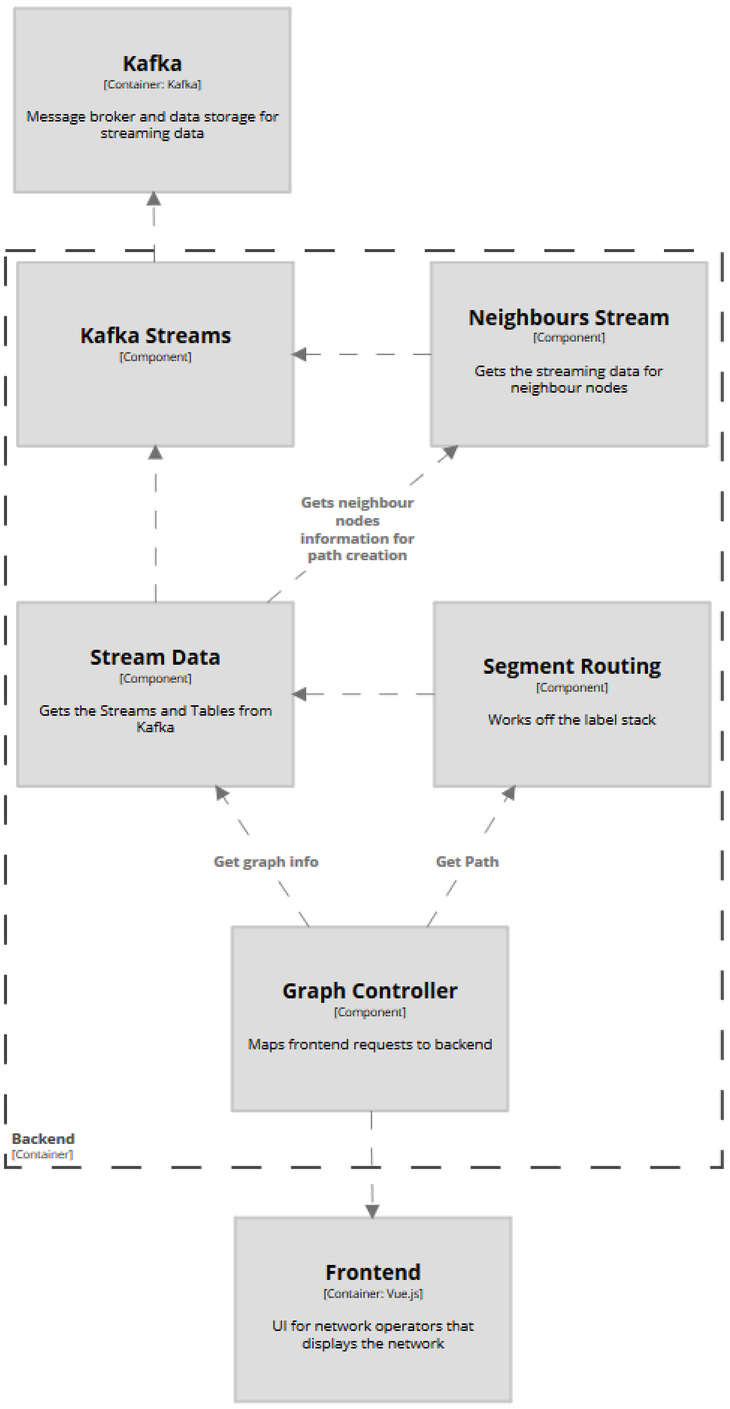

Kafka
[Container: Kafka]
Message broker and data storage for streaming data
Kafka Streams
[Component]
Neighbours Stream
[Component]
Gets the streaming data for neighbour nodes
Gets neighbour nodes information for path creation
Stream Data
[Component]
Gets the Streams and Tables from Kafka
Segment Routing
[Component]
Works off the label stack
Get graph info
Get Path
Graph Controller
[Component]
Maps frontend requests to backend
Backend
[Container]
Frontend
[Container: Vue.js]
UI for network operators that displays the network

Figure 6.7: Backend Component diagram

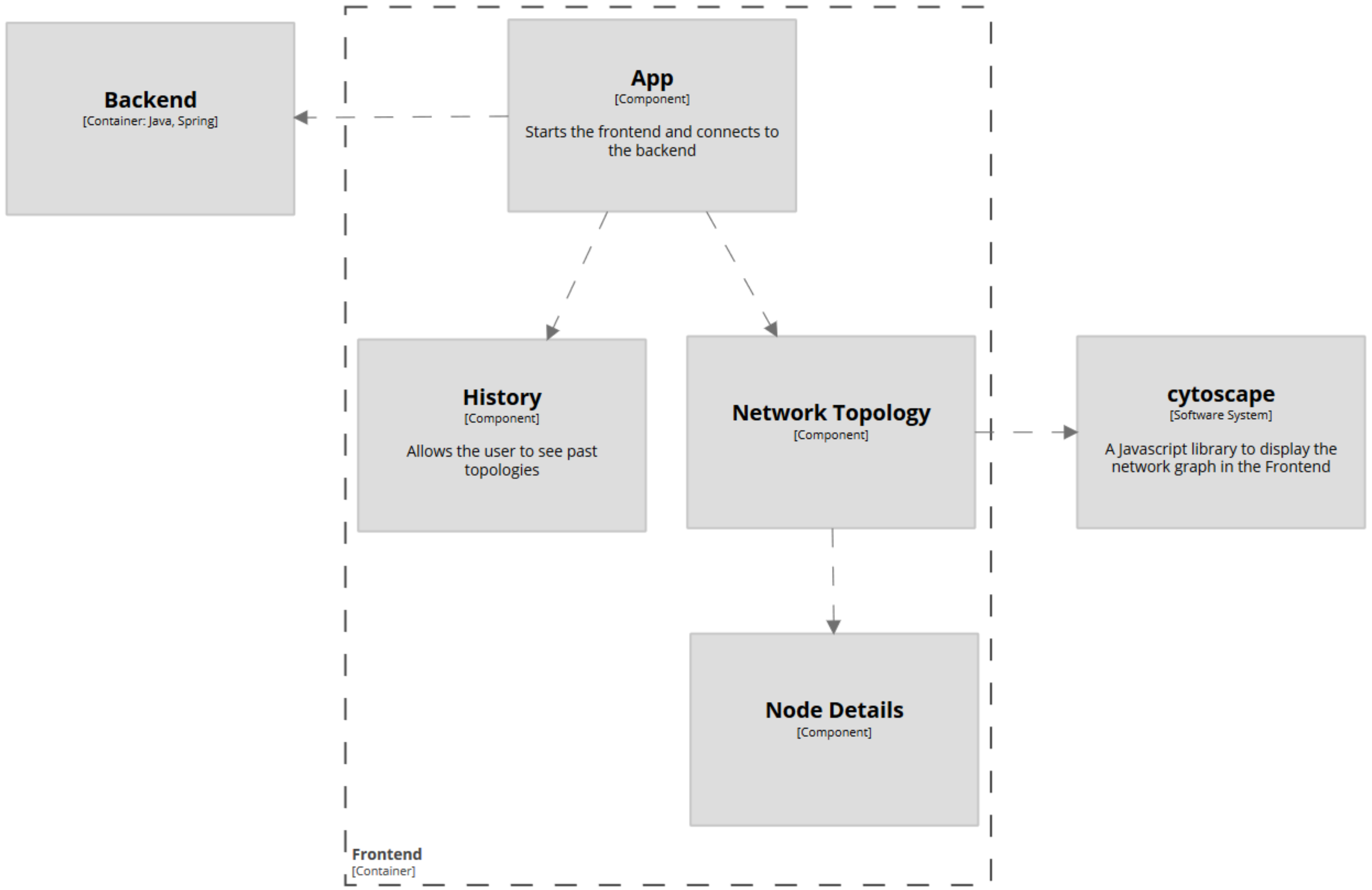

Backend
[Container: Java, Spring]
App
[Component]
Starts the frontend and connects to the backend
History
[Component]
Allows the user to see past topologies
Network Topology
[Component]
cytoscape
[Software System]
A Javascript library to display the network graph in the Frontend
Node Details
[Component]
Frontend
[Container]

Figure 6.8: Frontend Component diagram

6.4 Deployment

The application is deployed on a server provided by our advisor. All the repositories are containerized with docker for easy deployment and scalability of each component.

The client accesses the application via browser with HTTP requests (See 6.5).

The streaming data from the routers are sent via UDP to the application.

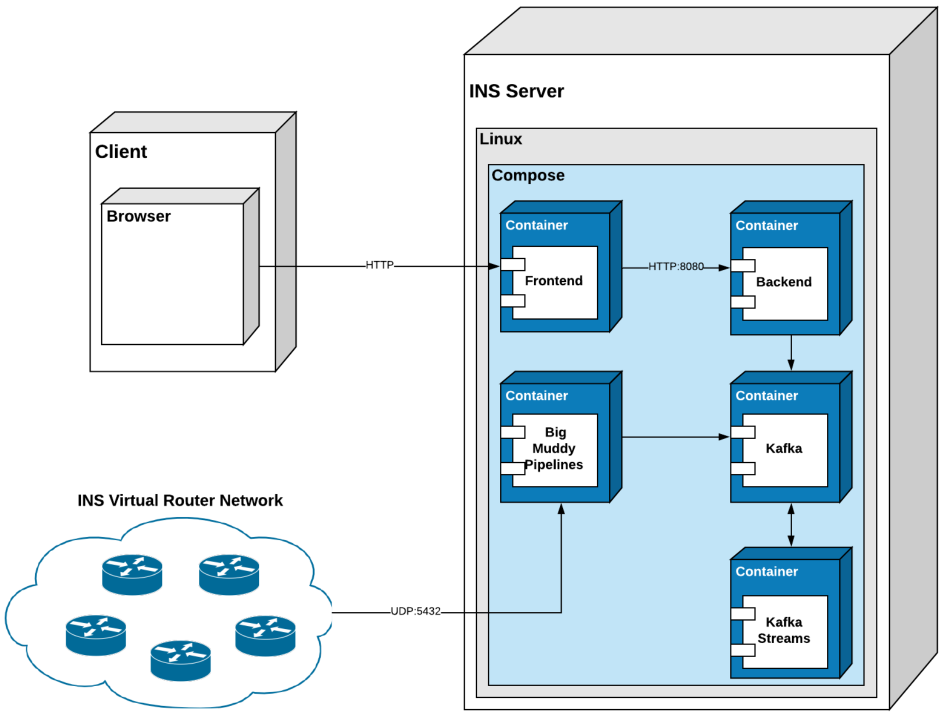
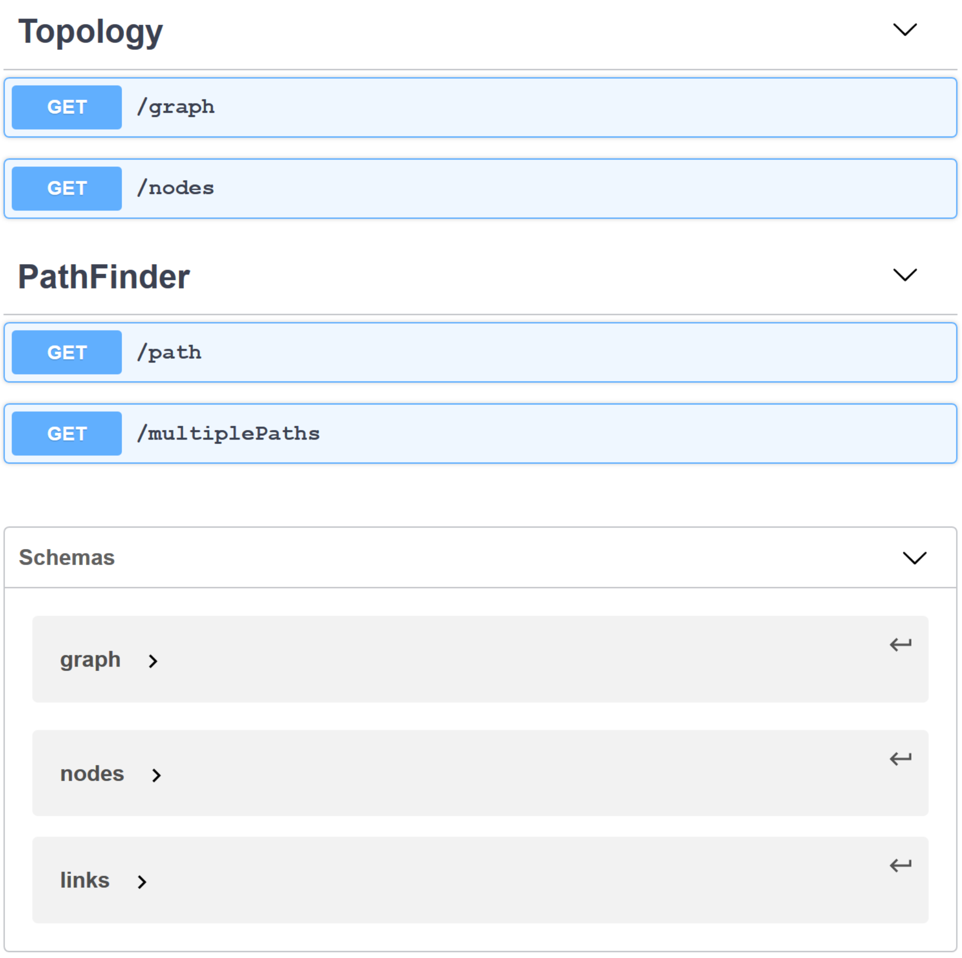


Figure 6.9: Deployment diagram

6.5 API

The REST API between backend and frontend is documented with the Swagger OpenAPI 3.0 specification. There are four GET methods on the interface, two of each for the topology and path finding.

- /graph
  get the graph layout of a network at a given time
- /nodes
  get node detail information of a network at a given time
- /path
  finds path from a source node to a destination node in a specific VRF
- /multiplePaths
  find all (sub) paths from a source node in a specific VRF


Topology

GET /graph

GET /nodes

PathFinder

GET /path

GET /multiplePaths

Schemas

graph

nodes

links

Figure 6.10: Swagger API

```yaml
openapi: 3.0.1
info:
  title: Streaming Telemetry Analytics
  description: Streaming Telemetry Analytics - get the graph layout of a
   ↪  router network and display paths in a vrf at a choosen time.
  version: '1.0'
servers:
  - url: 'http://10.20.0.10:8080'
paths:
  /graph:
    get:
      description: 'get the graph layout of a network at a given time (in
       ↪  ms)'
      tags:
        - Topology
      parameters:
        - in: query
          name: time
          required: true
          schema:
            type: integer
            example: 1559824800000
      responses:
        '200':
          description: return a list of nodes and links
          content:
            application/json; charset=utf-8:
              schema:
                $ref: '#/components/schemas/graph'
  /nodes:
    get:
      description: 'get node detail informations of a network at a given
       ↪  time (in ms)'
      tags:
        - Topology
      parameters:
        - in: query
          name: time
          required: true
          schema:
            type: integer
            example: 1559824800000
      responses:
        '200':
          description: return detail informations for all nodes
          content:
            application/json; charset=utf-8:
              schema:
                $ref: '#/components/schemas/nodes'
  /path:
    get:
      description: 'Find the path from a source node to a destination node in a specific VRF at a given time (in ms)'
      tags:
        - PathFinder
      parameters:
        - in: query
          name: departureNodeName
          required: true
          schema:
            type: string
            example: XR-01
        - in: query
          name: destinationNodeName
          required: true
          schema:
            type: string
            example: XR-09
        - in: query
          name: destinationGlobalIp
          required: true
          schema:
            type: string
            example: 9.9.9.9
        - in: query
          name: loopbackIp
          required: true
          schema:
            type: string
            example: 20.0.1.9
        - in: query
          name: time
          required: true
          schema:
            type: integer
            example: 1559824800000
      responses:
        '200':
          description: returns the found (sub) Path from source to destination node
          content:
            application/json; charset=utf-8:
              schema:
                $ref: '#/components/schemas/links'
  /multiplePaths:
    get:
      description: 'Find all (sub) paths from a source node in a specific VRF at a given time (in ms)'
      tags:
        - PathFinder
      parameters:
        - in: query
          name: departureNodeName
          required: true
          schema:
            type: string
            example: XR-01
        - in: query
          name: vrfName
          required: true
          schema:
            type: string
            example: Test
        - in: query
          name: time
          required: true
          schema:
            type: integer
            example: 1559824800000
      responses:
        '200':
          description: returns the found (sub) Paths from source node in the VRF
          content:
            application/json; charset=utf-8:
              schema:
                $ref: '#/components/schemas/links'
components:
  schemas:
    graph:
      type: object
      required:
        - nodes
        - links
      properties:
        nodes:
          type: array
          items:
            type: object
            properties:
              name:
                type: string
          example:
            - name: XR-01
            - name: XR-02
            - name: XR-03
        links:
          $ref: '#/components/schemas/links'
    nodes:
      type: object
      required:
        - nodesDetails
      properties:
        nodesDetails:
          type: array
          items:
            type: object
            properties:
              name:
                type: string
              primaryIP:
                type: string
              systemID:
                type: string
              label:
                type: string
              vrfs:
                type: array
                items:
                  type: object
                  properties:
                    name:
                      type: string
                    ip:
                      type: string
          example:
            - name: XR-01
              primaryIP: 1.1.1.1
              systemID: 0000.0000.1111
              label: 16001
              vrfs:
                - name: Test
                  id: 10.0.1.1
                - name: HSR-INS
                  id: 10.20.0.103
            - name: XR-02
              primaryIP: 2.2.2.2
              systemID: 0000.0000.2222
              label: 16002
              vrfs:
                - name: Test
                  id: 10.0.1.2
                - name: HSR-INS
                  id: 10.20.0.104
            - name: XR-03
              primaryIP: 3.3.3.3
              systemID: 0000.0000.3333
              label: 16003
              vrfs:
                - name: Test
                  id: 10.0.1.3
                - name: HSR-INS
                  id: 10.20.0.105
    links:
      type: array
      items:
        type: object
        properties:
          source:
            type: string
          target:
            type: string
      example:
        - source: XR-01
          target: XR-02
        - source: XR-02
          target: XR-03
        - source: XR-03
          target: XR-01
```

6.6 Sequence Diagrams

The shown sequence diagrams are an in-depth view of the previously documented API calls.

6.6.1 Get Graph and Get Nodes

The calls for to get the graph and the nodes could be refactored into one call in the future. Unfortunately there was not enough time to properly implement this.

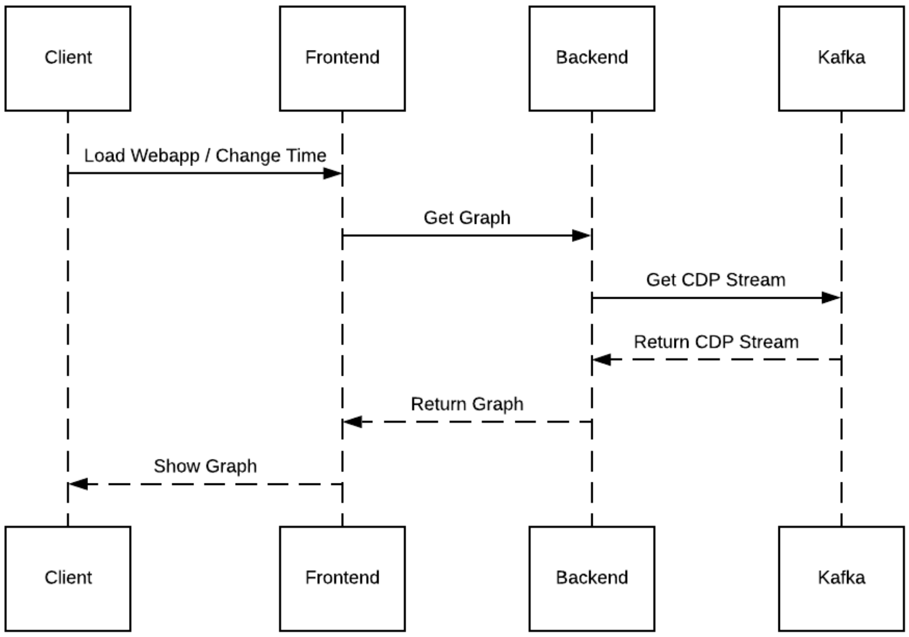


Figure 6.11: Sequence Diagram - Get Graph

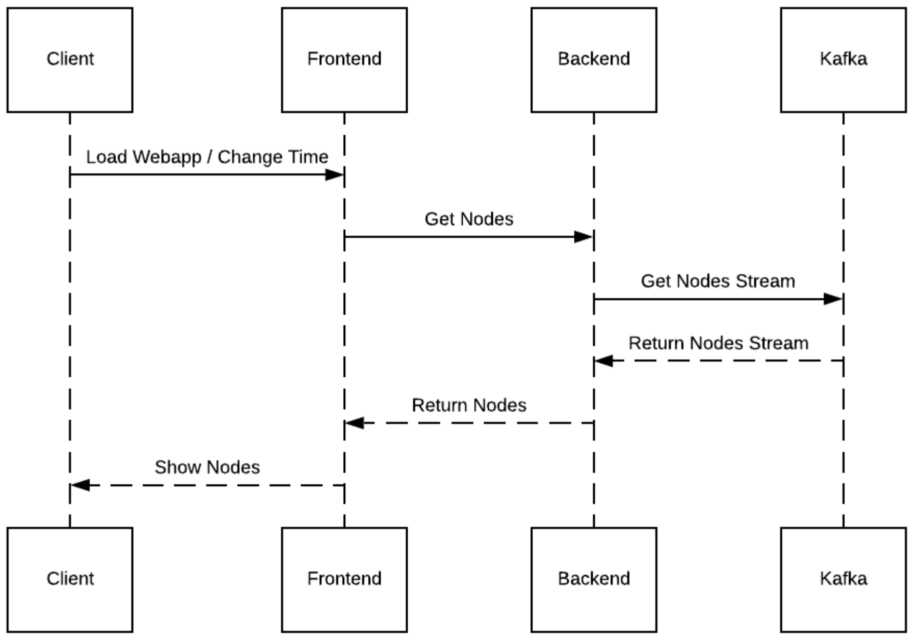


Figure 6.12: Sequence Diagram - Get Nodes

6.6.2 Get Multiple Paths and Get Path

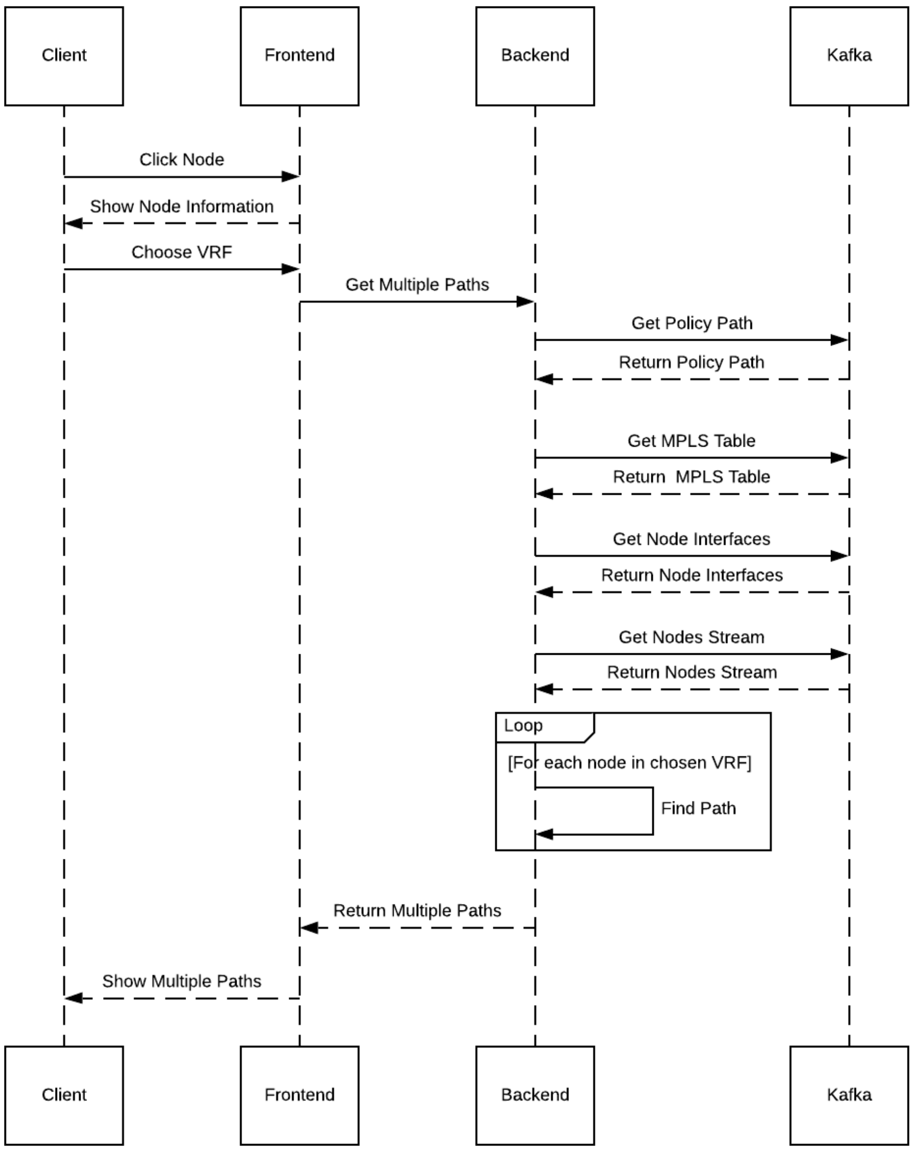


Figure 6.13: Sequence Diagram - Get Multiple Paths

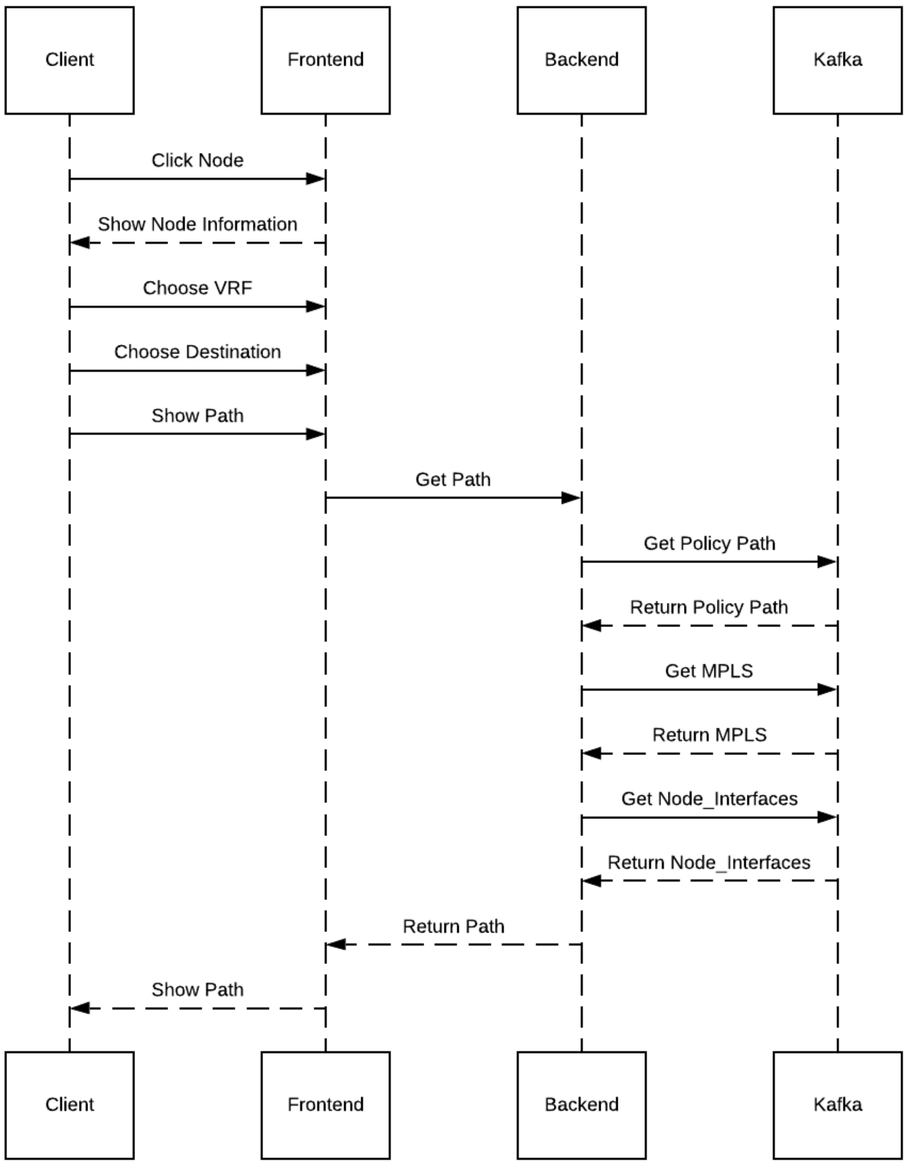


Figure 6.14: Sequence Diagram - Get Path

Chapter 7

Quality Risk Management

7.1 Quality Attributes

7.1.1 Performance

The user gets sub-second responses when operating with the application.

7.1.2 Usability

See the usability test in the appendix (A.3).

7.1.3 Modifiability

There are config-files in all components what makes them easily modifiable.

7.1.4 Reusability

Components are loosely coupled that makes them prone to change. For example the frontend can be easily changed, because it only uses the API from the backend.

7.1.5 Scalability

Docker sets a good foundation for the scalability of the application. The Kafka worker container could be scaled up easily so that there would be multiple Kafka instances to produce the pipeline data into Kafka streams.

The backend on the other hand does not need to be highly scalable, because there is no obvious scenario in which this is needed.

7.1.6 Security

Security has not been addressed as of now. In the future there are multiple possibility to implement security features:

- Authentication and Authorization (Checkpoint with RBAC, ACL)
- Communication encryption (SSL/TLS)

7.1.7 Reliability

With the help of Watchtower each container could be configured to reset a crashed container a predefined number of times.

7.2 Risk Management

The most dangerous risks for this project have been assessed and listed in the following Table. 7.1.

#	Description	Probability	Damage (h)	Mitigation	Measures
1	Loss of data due to technical errors or theft of Notebooks	0.05	8	"Commit early and often" Make Backups	Load Backup
2	Insufficient interfaces to surrounding systems (Libraries, Frameworks, Services) S	0.2	20	Evaluation of different systems.	Change system
3	Circumvention of team member	0.3	30	Eat healthy, get enough sleep and have fun.	Scale down scope
4	Insufficient communication	0.05	5	Talk early and often.	Communicate insufficient interactions.
5	Chosen graph library is missing a feature	0.2	40	Conduct a graph analysis	Change library.
6	Get lost in details. (Goldplating requirements)	0.1	20	Prioritize requirements and stick to the scope	Cancel requirements.
7	Missing or changed data from telemetry devices	0.15	30	Early testing against test data.	Code generically. Fake test data.

Table 7.1: Risk list

Chapter 8

Project Management

8.1 Organization

The project is run in a flat organization structure. All member are equal and are involved in the decision making process.

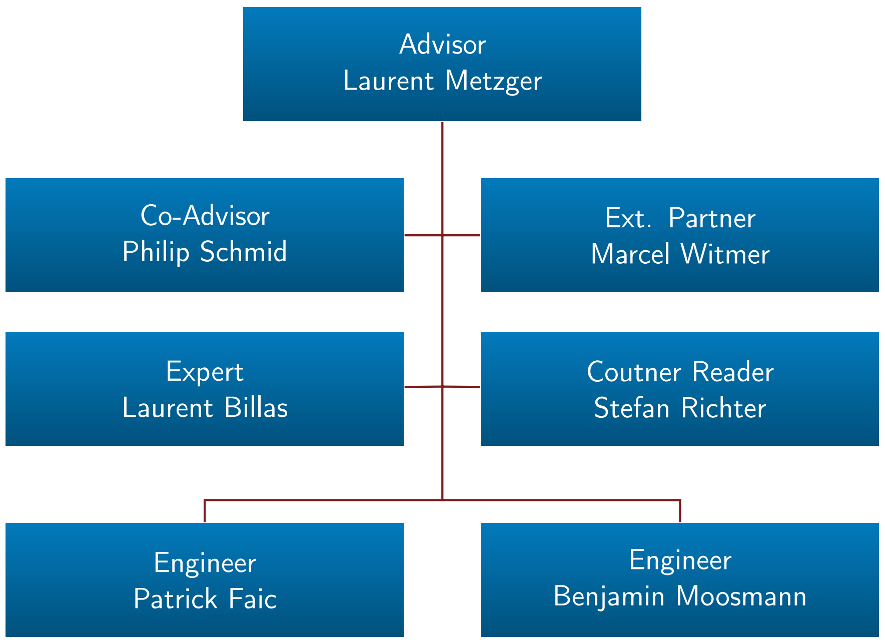
8.1.1 Members

Engineer	Patrick Faic	pfaic@hsr.ch
Engineer	Benjamin Moosmann	bmoosman@hsr.ch

8.1.2 External Contact Points

Advisor	Laurent Metzger	laurent.metzger@hsr.ch
Co-Advisor	Philip Schmid	philip.schmid@ins.hsr.ch
External Partner	Marcel Witmer	mawitmer@cisco.com

8.1.3 Diagram



8.2 Stakeholders

The following table 8.1 shows the stakeholders for this project. All the stakeholders have great influence in the decision making process what makes them "Key Players"

Stakeholder	Interests	Influence	Possible Conflicts
Advisor	Successful bachelor thesis, with an usable application, that may be extensible by future projects.	Defines scope and project goals in cooperation with Cisco.	Change the scope or requirements of the project.
Cisco	Usable and Appealing application.	Defines scope and project goals in cooperation with the advisor.	Change the scope or requirements of the project.
Dev-Team	Successful bachelor thesis, where all requirements are met and the partners are satisfied with.	Choice of technology within predefined borders.	Disputes over course of action.

Table 8.1: Stakeholders

8.3 Cost

The costs are taken directly from the "Bachelor Arbeit" module description.

Start:	18.02.2019
End:	14.06.2019
Weeks:	17
Weekly Cost:	20h + 2 * 45h in Week 16 and 17
Total Cost:	390h

Table 8.2: Project Cost

8.4 Phases and Milestones

Inception		
From	20.02.2019	
To	27.02.2019	
Milestones	Kickoff	20.02.2019

Elaboration		
From	27.02.2019	
To	13.03.2019	
Milestones	End of Elaboration	13.03.2019

Construction		
From	13.03.2019	
To	05.06.2019	
Milestones	Usability Testing	10.04.2019
	Demo	25.04.2019
	Feature Freeze	22.05.2019
	Code Freeze	05.06.2019

Transition		
From	05.06.2019	
To	14.06.2019	
Milestones	Delivery	14.06.2019

Table 8.3: Phases and Milestones

20.02	*Kickoff*
13.03	*EndofElaboration*
03.04	*UsabilityTesting*
25.04	*Demo*
22.05	*FeatureFreeze*
05.06	*CodeFreeze*
14.06	*Delivery*
05.07	*Presentation*

Note: The construction milestones (red) were shifted out of phase. The problem was a misunderstanding of critical data. The developers received wrong information about the labels needed for Segment Routing. (BGP-Labels instead of the used Policy-Labels) The result were wrongfully aggregated data and a loss of about 3 project-weeks to find and fix the error. The solution was to configure a different sensor path on the routers that contained the correct label information.

8.5 Work Flow

Scrum + Unified Process are used as the agile project management framework. Due to the small developer size a daily stand-up is not needed.

Sprint duration: 1 Week (Wednesday to Wednesday)
Effort Estimation: Planning Poker (Values: 1, 2, 3, 5, 8, 13, 20)
Sprint Review Time Frame: 10 minutes.

8.5.1 Time and Issue Tracking

Jira is used for time and issue tracking. (For details see chapter E the resulting Charts in the appendix.)

8.5.2 Definition of Done

For coding related issues in the sprint backlog to be "Done", the following points have to be fulfilled:

- Code Style Guidelines followed
- Code compiles (0 Errors, 0 Warnings)
- Code reviewed/Pair-Programming
- Code committed
- Build faultless
- Documentation updated

8.5.3 End of Sprint

After each meeting the following steps occur:

- Review active sprint.
- Retrospective of active sprint.
- Close active sprint.
- Start new sprint.

8.6 Infrastructure

8.6.1 Version Control and Continuous Integration

GitLab is used for version control and CI/CD.

Repositories:

- https://gitlab.com/bartbeni/ba-seg-routing-kafka-workers
- https://gitlab.com/bartbeni/ba-seg-routing-frontend
- https://gitlab.com/bartbeni/ba-seg-routing-backend
- https://gitlab.com/bartbeni/ba-seg-routing-streaming-telemetry

8.6.2 Toolchain

The following tools are used in this project:

For each repository the code is pushed to GitLab, where the app is build and the docker container is created and pushed to the GitLab registy.

Watchtower is a tool recommended from Philip Schmid. It gets the images from the GitLab Registry and deploys the containers. On completion Watchtower sends the status to a arbitrary messenger.

Figure 8.1 gives a good overview of the toolchain.

Use	Tool
Frontend	Node (9.9.0)
Frontend Framework	Vue (2.5.17)
Topology Library	vue-cytoscape (0.2.8), tippy.js (4.2.0)
Backend	Java (12.0.1)
Data Streaming	Kafka (5.0.0)
Container	Docker
Version Control	GitLab
Mockups	Balsamiq
Diagrams	Lucidchart, Structurizer
Documentation	Overleaf

Table 8.4: Tools

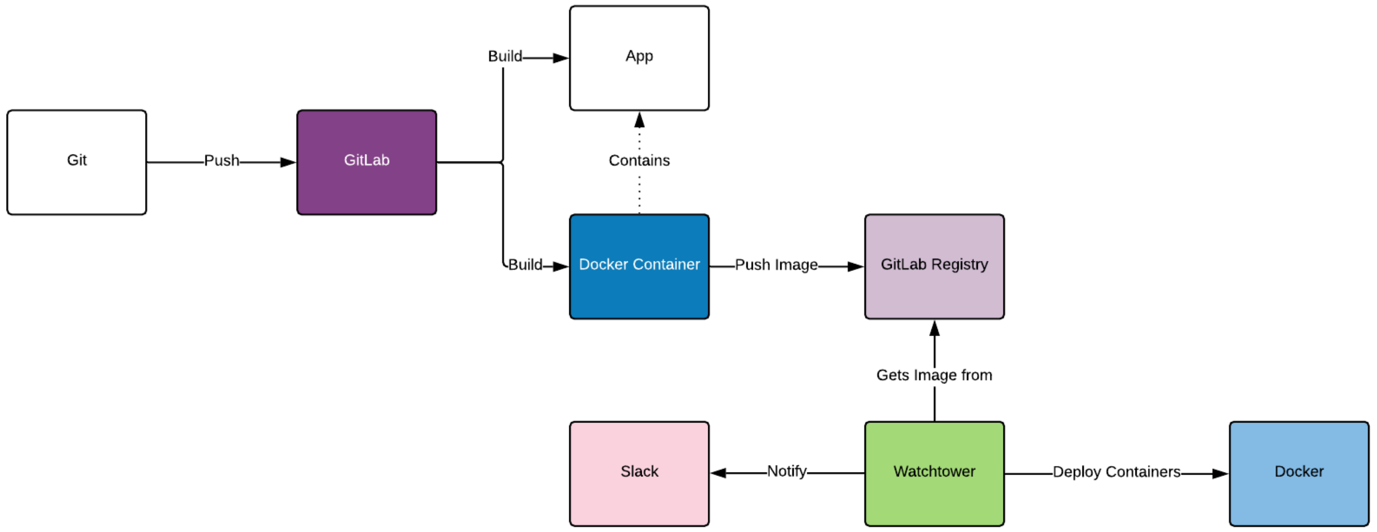


Figure 8.1: Toolchain

8.7 Meetings

Meetings are held each Wednesday at 9:00. (For details see chapter F in the appendix.)

Part III

Appendix

Appendix A

Test Report

A.1 Unit- and Integration-Tests

The coupling between the business logic and Kafka in this application is really strong. If the Kafka involvements would have been mocked, there would be nearly no logic left for testing.
Integration-Tests to verify the interaction between Kafka and the rest of the application were intended but due to errors from Kafka during development they could not be achieved.
Because of this reasons there were only manual testing of the application.

A.2 System Tests

ID	ST-1 - Network Topology Displayed in Frontend
Test Goal	The Application receives Streaming Telemetry Data from the Pipeline and displays all the existing nodes and links.
Precondition	• All docker container are running • Logged in to the INS network
Steps	1. Open browser 2. Open 10.20.0.10
Expectation	Network topology displayed

Result	Successful
Covered UCs	UC-1, UC-2, UC-3

Table A.1: ST-1

ID	ST-2 - Display Graph History
Test Goals	• When the date or time is changed, the topology of that time is shown. • Future dates or times can not be chosen.
Precondition	ST-1
Steps	Choose a day or time in the past with the slider or date picker.
Expectation	The topology of the chosen time is shown.
Result	Successful
Covered UCs	UC-4

Table A.2: ST-2

ID	ST-3 - Display Router Details
Test Goals	A click on a network node opens a tool-tip with detailed information and selection items.
Precondition	ST-1
Steps	Click on a network node.
Expectation	Detailed information is shown in a tool-tip.
Result	Successful
Covered UCs	UC-8

Table A.3: ST-3

ID	ST-4 - Display Path
Test Goals	Display a Segment Routing path between two nodes.
Precondition	ST-1, ST-3
Steps	1. Click on a node (source) 2. Choose a VRF 3. Choose a destination 4. Click show Path
Expectation	A highlighted path between the source node and destination
Result	Successful
Covered UCs	UC-6,

Table A.4: ST-4

A.3 Usability Tests

To ensure a user-friendly application, usability tests are conducted with 4 test persons.

A.3.1 Subjects

The target groups for this application are network operators and computer scientists. We focused on participants with experience in the network domain.

A.3.2 Pre-Interview

- Are you familiar with network technologies concepts?
- How do you usually check the status of a network?
- Have you ever used a graphical UI for network management?

A.3.3 Process

You are the operator of a router network. The status of the network you see is the expected working network.

- What is the primary IP of the router XR-08?
- What is the status of the network at the 10th of June at 12:00? Are there any changes to the initial network?
- What is the status of the network at the 10th of June at 12:10? Are there any changes to the initial network?
- What are the neighbours of router XR-04?
- What are all the possible routes from XR-09 in the "Test" VRF?
- What is the actual route of the "Test" VRF From router XR-09 to XR-01?
- What is the actual route of the "SA" VRF From router XR-09 to XR-01?

A.3.4 Post-Interview

- What was confusing/unclear?
- What did you like about the application?
- What other functionality would you like to see?
- Where do you see room for improvement?

A.3.5 Findings

Time Slider

The time slider is sometimes not used for small time steps. The date-time-picker is preferred.

Zoom

The zoom needs to be regulated with a min/max value because it happened that the topology disappeared due to extensive zooming out.

Visualization

The visualization and coloring of the UI is considered clean and clear.

Tool-tip

- Deselect tool-tip with a second click on the selected node.
- tool-tip is not disappearing when date-time-picker is pressed.

Additional Functionality

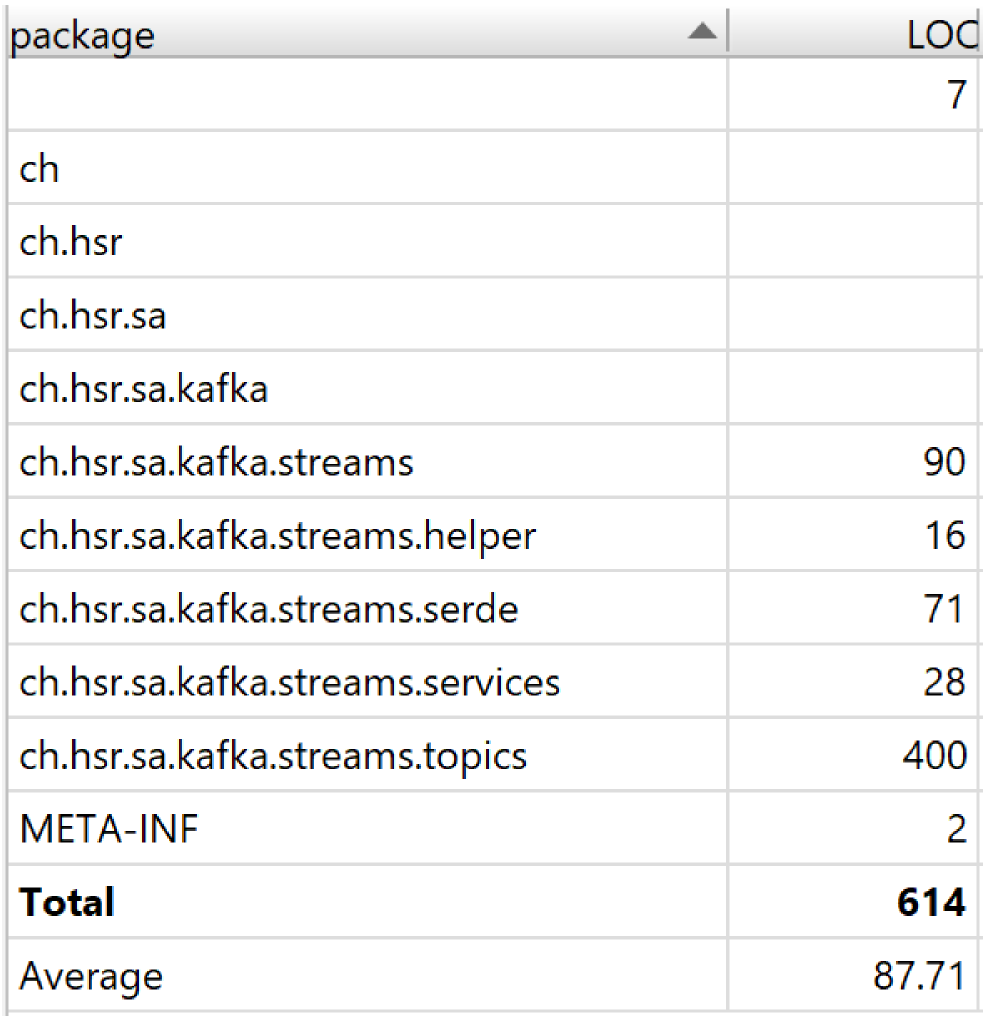
- Show Ethernet ports between routers.
- Show additional routing table information.

Appendix B

Metrics

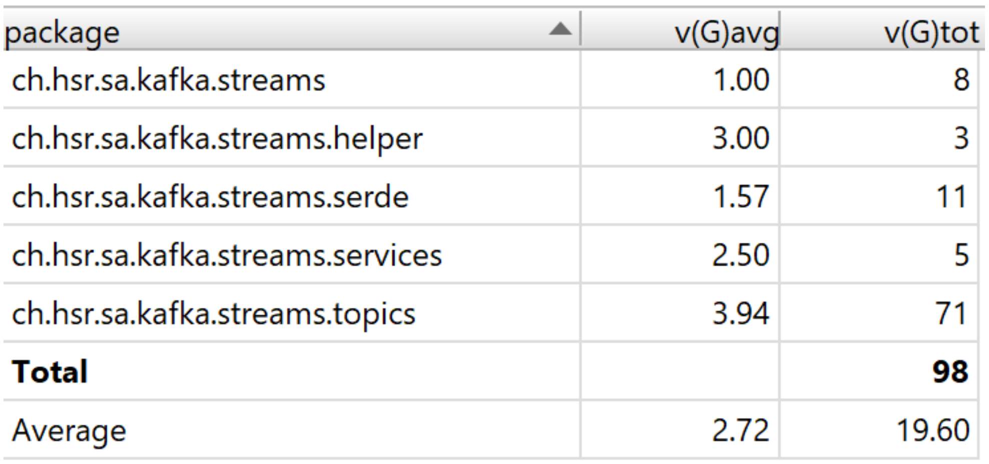
LOC	Lines of Code
v(G)avg	Average Cyclomatic Complexity
v(G)tot	Total Cyclomatic Complexity
OCavg	Average Operation Complexity
WMC	Weighted Method Complexity

B.1 Kafka Worker



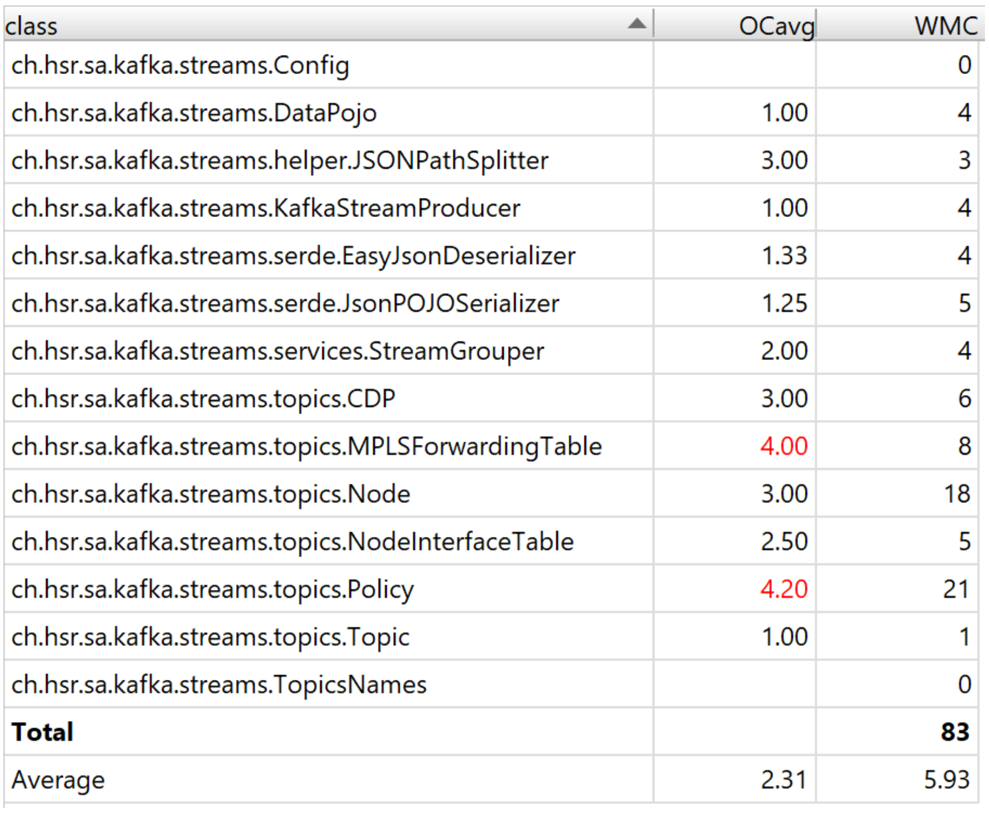
package	LOC
	7
ch	
ch.hsr	
ch.hsr.sa	
ch.hsr.sa.kafka	
ch.hsr.sa.kafka.streams	90
ch.hsr.sa.kafka.streams.helper	16
ch.hsr.sa.kafka.streams.serde	71
ch.hsr.sa.kafka.streams.services	28
ch.hsr.sa.kafka.streams.topics	400
META-INF	2
Total	**614**
Average	87.71

Figure B.1: Kafka Worker Package Lines of Code (LOC)



package	v(G)avg	v(G)tot
ch.hsr.sa.kafka.streams	1.00	8
ch.hsr.sa.kafka.streams.helper	3.00	3
ch.hsr.sa.kafka.streams.serde	1.57	11
ch.hsr.sa.kafka.streams.services	2.50	5
ch.hsr.sa.kafka.streams.topics	3.94	71
Total		**98**
Average	2.72	19.60

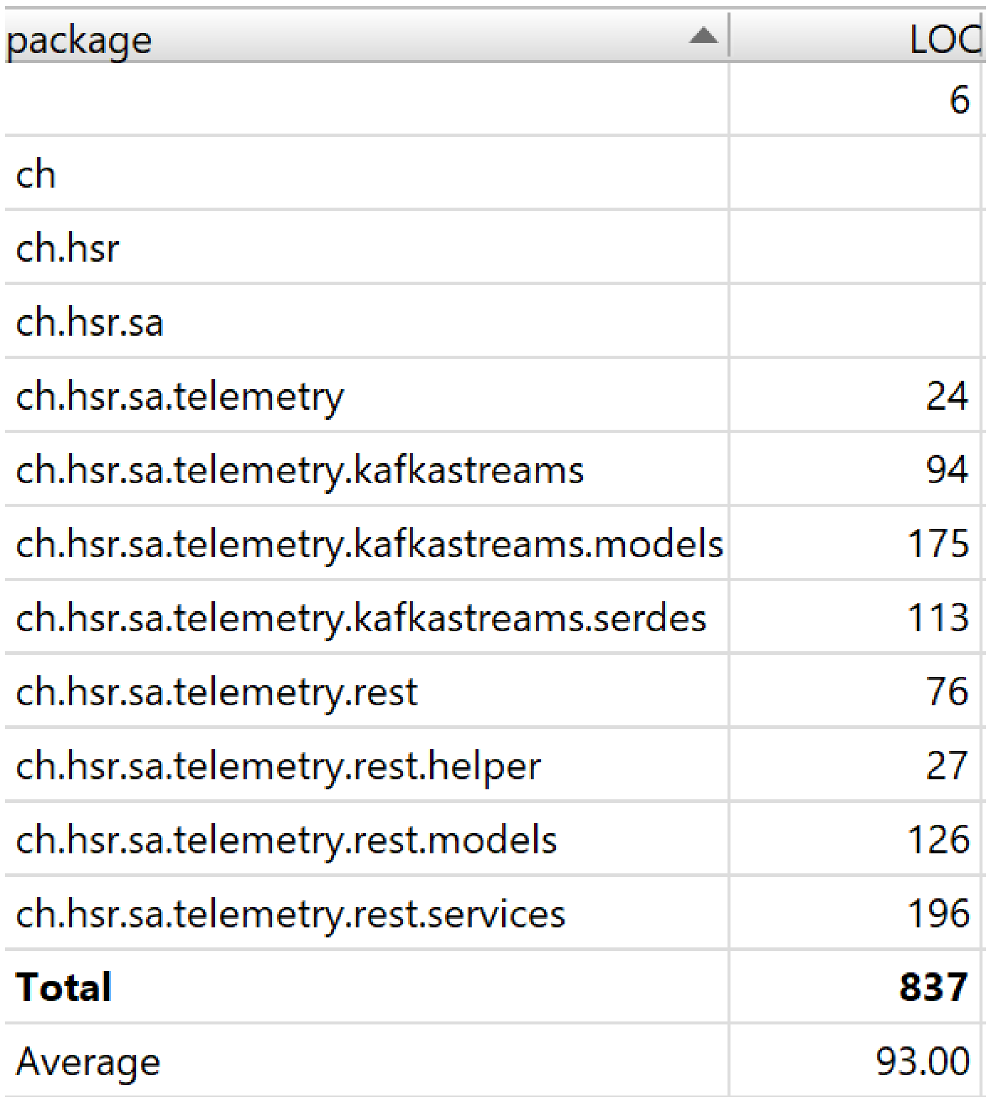
Figure B.2: Kafka Worker Package Complexity



class	OCavg	WMC
ch.hsr.sa.kafka.streams.Config		0
ch.hsr.sa.kafka.streams.DataPojo	1.00	4
ch.hsr.sa.kafka.streams.helper.JSONPathSplitter	3.00	3
ch.hsr.sa.kafka.streams.KafkaStreamProducer	1.00	4
ch.hsr.sa.kafka.streams.serde.EasyJsonDeserializer	1.33	4
ch.hsr.sa.kafka.streams.serde.JsonPOJOSerializer	1.25	5
ch.hsr.sa.kafka.streams.services.StreamGrouper	2.00	4
ch.hsr.sa.kafka.streams.topics.CDP	3.00	6
ch.hsr.sa.kafka.streams.topics.MPLSForwardingTable	4.00	8
ch.hsr.sa.kafka.streams.topics.Node	3.00	18
ch.hsr.sa.kafka.streams.topics.NodeInterfaceTable	2.50	5
ch.hsr.sa.kafka.streams.topics.Policy	4.20	21
ch.hsr.sa.kafka.streams.topics.Topic	1.00	1
ch.hsr.sa.kafka.streams.TopicsNames		0
Total		**83**
Average	2.31	5.93

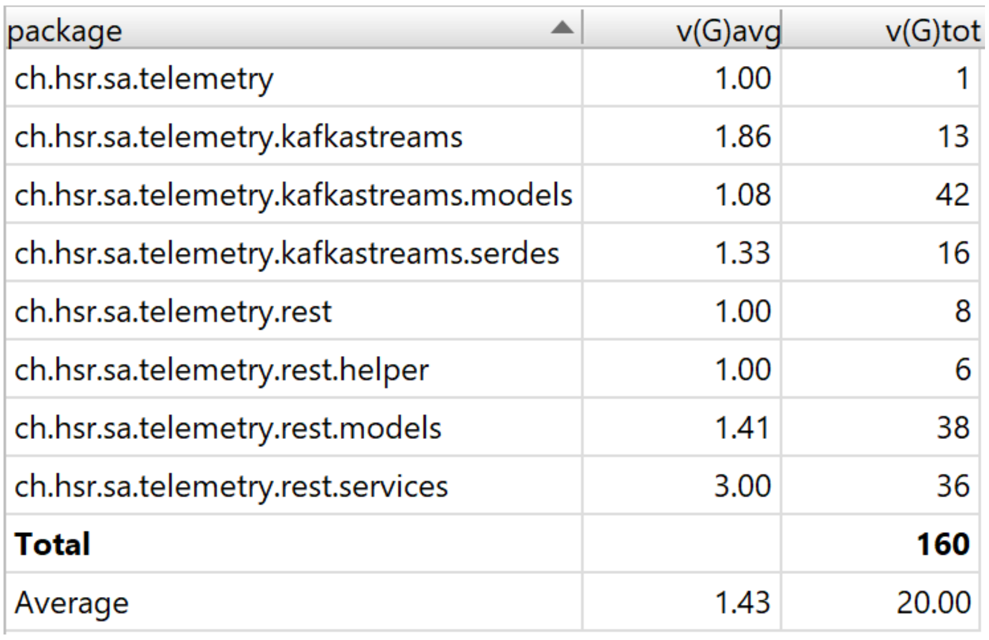
Figure B.3: Kafka Worker Class Complexity

B.2 Backend



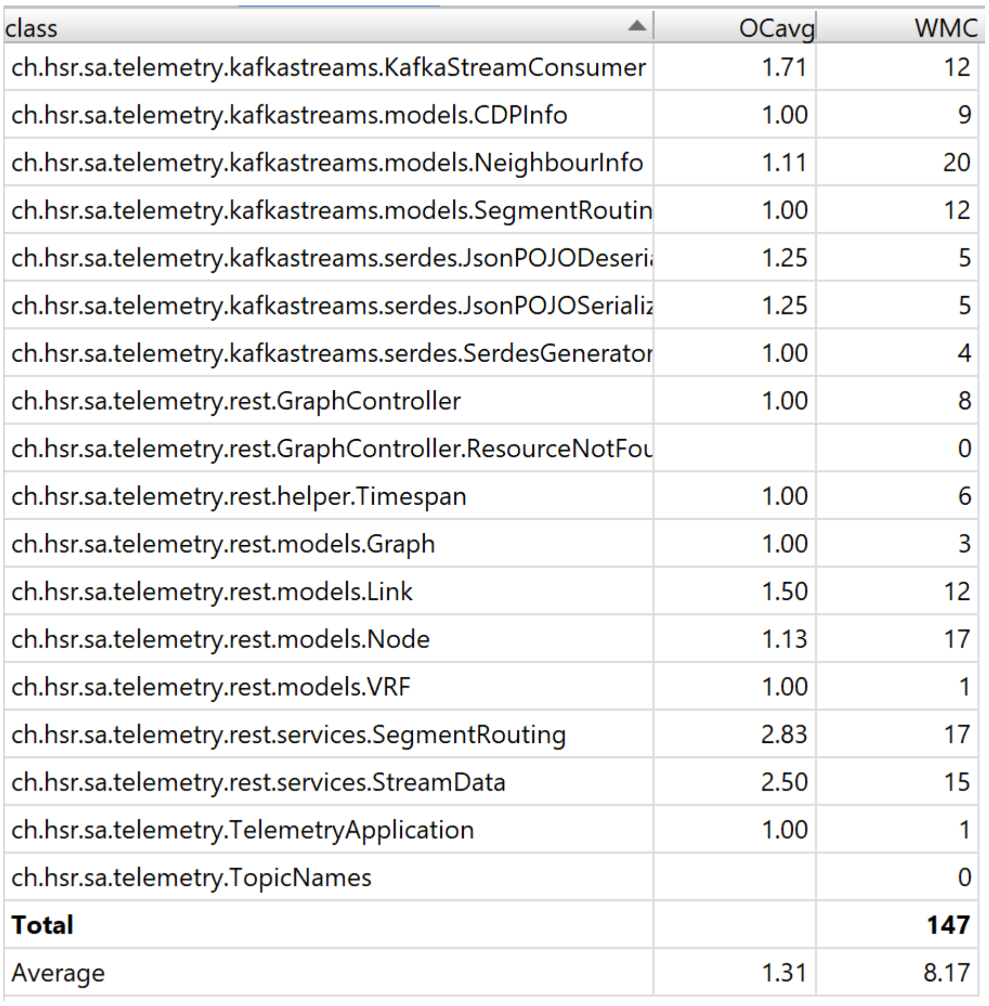
package	LOC
	6
ch	
ch.hsr	
ch.hsr.sa	
ch.hsr.sa.telemetry	24
ch.hsr.sa.telemetry.kafkastreams	94
ch.hsr.sa.telemetry.kafkastreams.models	175
ch.hsr.sa.telemetry.kafkastreams.serdes	113
ch.hsr.sa.telemetry.rest	76
ch.hsr.sa.telemetry.rest.helper	27
ch.hsr.sa.telemetry.rest.models	126
ch.hsr.sa.telemetry.rest.services	196
Total	**837**
Average	93.00

Figure B.4: Backend Package Lines of Code (LOC)



package	v(G)avg	v(G)tot
ch.hsr.sa.telemetry	1.00	1
ch.hsr.sa.telemetry.kafkastreams	1.86	13
ch.hsr.sa.telemetry.kafkastreams.models	1.08	42
ch.hsr.sa.telemetry.kafkastreams.serdes	1.33	16
ch.hsr.sa.telemetry.rest	1.00	8
ch.hsr.sa.telemetry.rest.helper	1.00	6
ch.hsr.sa.telemetry.rest.models	1.41	38
ch.hsr.sa.telemetry.rest.services	3.00	36
Total		**160**
Average	1.43	20.00

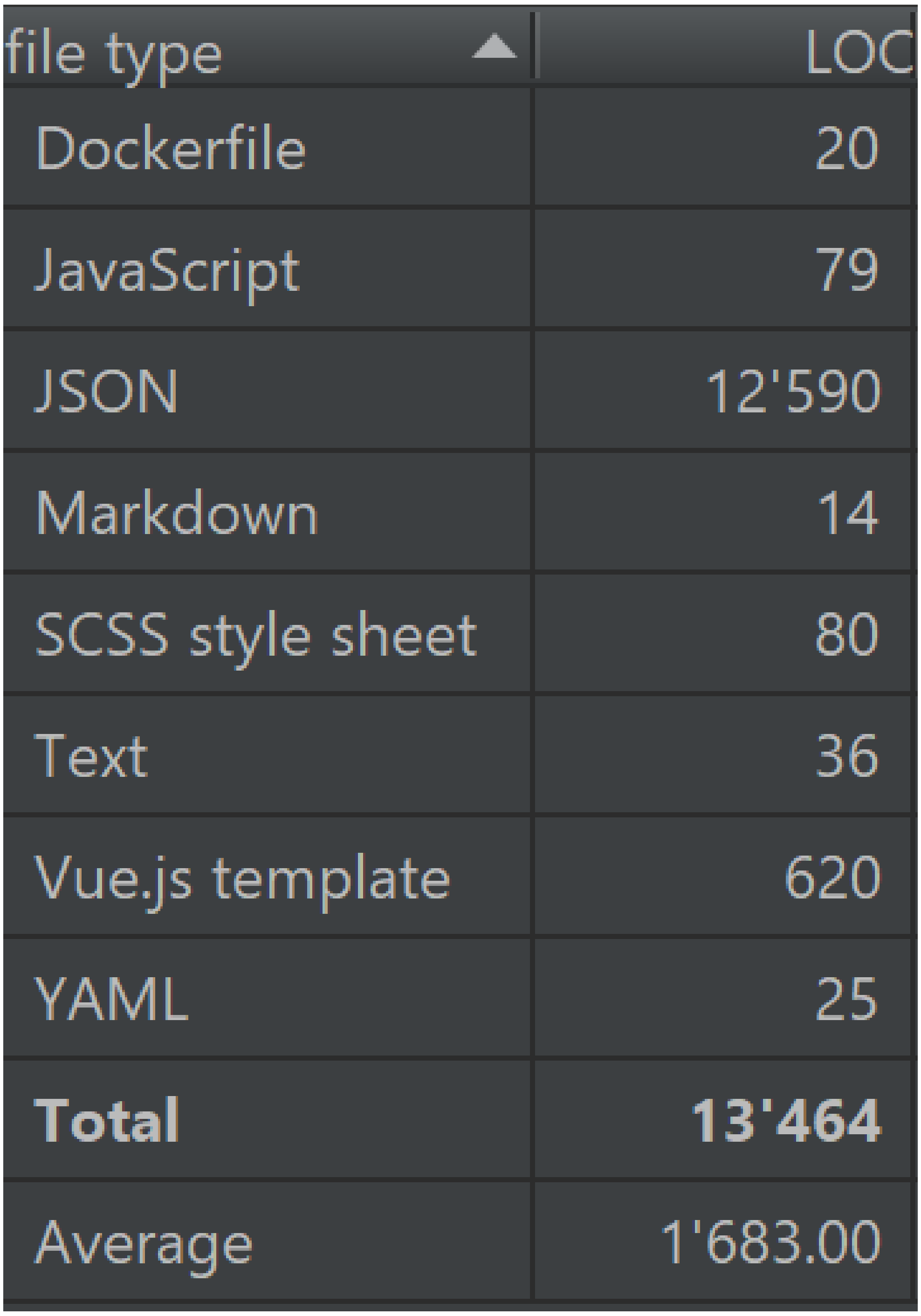
Figure B.5: Backend Package Complexity



class	OCavg	WMC
ch.hsr.sa.telemetry.kafkastreams.KafkaStreamConsumer	1.71	12
ch.hsr.sa.telemetry.kafkastreams.models.CDPInfo	1.00	9
ch.hsr.sa.telemetry.kafkastreams.models.NeighbourInfo	1.11	20
ch.hsr.sa.telemetry.kafkastreams.models.SegmentRoutin	1.00	12
ch.hsr.sa.telemetry.kafkastreams.serdes.JsonPOJODeseri	1.25	5
ch.hsr.sa.telemetry.kafkastreams.serdes.JsonPOJOSerializ	1.25	5
ch.hsr.sa.telemetry.kafkastreams.serdes.SerdesGenerator	1.00	4
ch.hsr.sa.telemetry.rest.GraphController	1.00	8
ch.hsr.sa.telemetry.rest.GraphController.ResourceNotFou		0
ch.hsr.sa.telemetry.rest.helper.Timespan	1.00	6
ch.hsr.sa.telemetry.rest.models.Graph	1.00	3
ch.hsr.sa.telemetry.rest.models.Link	1.50	12
ch.hsr.sa.telemetry.rest.models.Node	1.13	17
ch.hsr.sa.telemetry.rest.models.VRF	1.00	1
ch.hsr.sa.telemetry.rest.services.SegmentRouting	2.83	17
ch.hsr.sa.telemetry.rest.services.StreamData	2.50	15
ch.hsr.sa.telemetry.TelemetryApplication	1.00	1
ch.hsr.sa.telemetry.TopicNames		0
Total		**147**
Average	1.31	8.17

Figure B.6: Backend Class Complexity

B.3 Frontend



file type	LOC
Dockerfile	20
JavaScript	79
JSON	12'590
Markdown	14
SCSS style sheet	80
Text	36
Vue.js template	620
YAML	25
Total	**13'464**
Average	1'683.00

Figure B.7: Frontend Package Lines of Code (LOC)

Appendix C

Manual

C.1 Installation Guideline

C.1.1 Frontend

To run the Frontend locally use `npm run serve`

C.1.2 Backend

When the Backend is run locally, make sure that the the application id under src/main/resources/application.properties is different from the deployed application id.

```
spring.kafka.streams.application-id=<my-id>
```

otherwise you will get an error:

"The state store, <my-topic>, may have migrated to another instance."

C.1.3 Server

Putty

To access the server putty is needed. Follow these steps:

1. Download Putty (`https://putty.org/`)
2. Download PuTTYgen (`https://www.ssh.com/ssh/putty/windows/puttygen`)
3. Download WinSCP (`https://winscp.net/eng/index.php`)
4. In PuTTYgen: load existing private key created for the server and save generated public and private keys.
5. In Putty Session: Set Host Name or IP address
6. In Putty Connection/Data: Set Auto-log-in user name to "ins"

7. In Putty Connection/SSH/Auth: Load your generated private Key.
8. In Putty Session: Save Config.
9. In Putty Session: Load and Open a saved session.

Docker Commands

Once on the server command line, docker commands can be helpful:

Show all docker containers

```
docker ps -a
```

Start docker container

```
docker start <container-id>
```

Stop docker container

```
docker stop <container-id>
```

Execute commands on the container.

```
docker exec -it <container-id> /bin/bash
```

Update all images from the docker-compose file. (first navigate to the folder with the docker-compose)

```
cd ba-seg-routing-streaming-telemetry

docker-compose up -d
```

Dump a specified topic into a text file.

```
docker run --net=host --rm confluentinc/cp-kafka:5.0.0 kafka-console-consumer
    --bootstrap-server localhost:29092 --topic <my-Topic-name> --property
    print.key=true > <file-name>.txt
```

Dump the last 1000 (offset) directly into the console (no "> <file-name>.txt)

```
docker run --net=host --rm confluentinc/cp-kafka:5.0.0 kafka-console-consumer
    --bootstrap-server localhost:29092 --topic telemetry --property
    print.key=true --offset 1000
```

C.2 Debugging

C.2.1 Kafka-Workers

If the app is run in debug mode it can take up to 5 minutes until a break point will trigger, due to the interval of data send.

C.2.2 Console Debugging

Run the application locally and print to the console.

C.2.3 API Debugging

Define APIs in the Backend to receive the data requested in the browser. For example:

```
@RequestMapping("debugging/path/mpls")
public HashMap<String, String> mplsTable(@RequestParam(value = "time") long
     currentTimeInMs){
    timespan.setTimespan(currentTimeInMs);
    return
        streamData.getKafkaTable(TopicNames.MPLS_FORWARDING_TABLE_WINDOWED,
        timespan);
}
```

C.2.4 Topic Debugging

For analyzing a KStream, create a new topic in the Kafka-Worker-Repository:

```
<myStream>.to("<topicName>", Produced.with(Serdes.String(),
    telemetryDataSerde))
```

Note: Serdes.String() for String, telemetryDataSerde for JsonNode.

You can then dump the topic and analyze the stream.

Dump a specified topic into a text file.

```
docker run --net=host --rm confluentinc/cp-kafka:5.0.0 kafka-console-consumer
    --bootstrap-server localhost:29092 --topic <my-Topic-name> --property
    print.key=true > <file-name>.txt
```

C.3 Known Problems

This section elaborates on problems who were encountered during the project and possible solutions.

C.3.1 Using an application ID that is already used

Problem: During development, if you want to run the backend locally the following error (internal server error 500) occurs:

"The state store, <my-topic>, may have migrated to another instance."

Solution: In the backend under src/main/resources/application.properties change the id:

```
spring.kafka.streams.application-id=<my-id>
```

This is a problem because the consumer's id needs to be unique in order to receive all messages of a topic.

C.3.2 Full Kafka Volume

Problem: After a while the Kafka volume is full and the container will crash.

Solution: Free up space by deleting data. Use the following command to delete a Kafka topic inside a container. It is not recommended to delete the topic "telemetry".

This error was of too many records logged by the software.

```
./bin/kafka-topics.sh --zookeeper localhost:2181 --delete --topic <my-topic>
```

Alternatively remove everything inside opt data:

```
rm -rf /opt/data/kafka-data/*}
```

C.3.3 DirectoryNotEmptyException

Problem: when running the Kafka-stream-worker repository locally, Kafka makes a temporary folder with telemetry filter tests. When the repository is rerun it is possible that a DirectoryNotEmptyException is thrown.

Solution: Delete the telemetry-filter-tests folder in $D : \backslash tmp \backslash kafka - streams$ (D = Drive where the repository is stored)

C.4 Gitlab Readme

C.4.1 Streaming-Telemetry

This project is part of a Bachelor-Thesis.
The goal of the application is to display a network and its dynamic routing paths in nearly real-time using telemetry data send by the routers. The network is built by consuming and aggregating CDP information. VRF, MPLS and IPV4 network information consumed used and transformed, to display a routing path between nodes. The data flow is from the Telemetry Devices over Cisco Big Muddy Pipeline to Kafka.From there a Kafka worker produces topics which are consumed by a Java Spring Boot Backend Application. A Vue.js Web-Application gets the network and path information from the backend and visualise them in a cytoscape graph.

Part of the whole application are following projects:

- ba-seg-routing-streaming-telemetry
- ba-seg-routing-kafka-workers
- ba-seg-routing-backend
- ba-seg-routing-frontend

Initial Deployment of all Applications

Dependencies

- Git
- Docker
- Access to Gitlab and Dockerhub repositories

First Deployment
install git on server
install Docker on server

```
docker login registry.gitlab.com
git pull ba-seg-routing-streaming-telemetry
cd ba-seg-routing-streaming-telemetry
docker-compose up -d
```

Install Watchtower:

```
cd ba-seg-routing-streaming-telemetry/watchtower
```

change in *private-watchtower.env* the web hook url for your slack bot

```
WATCHTOWER_NOTIFICATION_SLACK_HOOK_URL=<your-slack-web-hook-url>
docker-compose up -d
```

Now new images on gitlab registry will be pulled and automatically deployed. You get notifications of the deployment in your slack channel.

Deploy Code

All projects have an *.gitlab-ci.yml* which creates a new Docker image at the Gitlab registry. If Watchtower is installed, it will deploy the latest image.

```
docker push registry.gitlab.com/<username>/<project-name>:latest
```

Without Watchtower you can use the *update.sh*script to update all containers:

```
docker-compose rm -s -f
docker-compose pull
docker-compose up -d
```

Or use the same commands with a '[SERVICE...]' statement on the end of each line to only update a specific container.

Reset Stream

To reset the stream and let the application reprocess the history:

```
docker run
  -p 29092:29092
  --rm confluentinc/cp-kafka:5.0.0
  kafka-streams-application-reset --application-id <application-id>
      --input-topics telemetry-cdp-grouped --bootstrap-servers
      <hostname>:29092 --zookeeper <hostname>:32181
```

C.4.2 BA Kafka Worker

This project produces Kafka topics from the Bigmuddy Pipeline telemetry stream.
The following topics are produced:

- telemetry-mpls-table
- telemetry-node-interface-table
- telemetry-cdp-grouped
- telemetry-nodes
- telemetry-sr-path

Information to the whole Application

Streaming-Telemetry: Installation Guide and more Information

Config

The Configuration for the Kafka instance can be found here:
telemetryfilter/src/main/java/ch/hsr/sa/kafka/streams/Config.java

```
public static final String APP_NAME = "telemetry-filter";
public static final String BOOTSTRAP_SERVERS = "10.20.0.10:29092";
public static final String AUTO_OFFSET_RESET = "earliest";
```

C.4.3 BA Backend

This Java Spring Boot project consumes Kafka topics and process those, for requesting a network of routers and paths between them, as web calls.

Information to the whole Application

Streaming-Telemetry: Installation Guide and more Information

Config

src/main/resources/application.properties

```
spring.kafka.bootstrap-servers=10.20.0.10:29092
spring.kafka.streams.application-id=telemetry-backend
spring.kafka.consumer.auto-offset-reset=earliest
spring.kafka.consumer.value-deserializer=org.springframework.kafka.support.
    serializer.JsonDeserializer
spring.kafka.consumer.properties.spring.json.trusted.packages=sample.kafka
spring.kafka.producer.value-serializer=org.springframework.kafka.support.
    serializer.JsonSerializer
```

Note: make sure the application-id is unique

Swagger API

openapi.yaml

C.4.4 BA Frontend

This Vue.js project displays the network with nodes and links in a Cytoscape graph.
When selecting a node you can display the path in a VRF to a destination node or see all paths from this node in a VRF.

Information to the whole Application

Streaming-Telemetry: Installation Guide and more Information

Config

The Configuration for the backend path can be found here:
.env

```
VUE_APP_BACKEND_PATH=http://10.20.0.10:8080/
```

Run locally

```
npm run serve
```

Appendix D

Personal Reports

D.1 Patrick Faic

Dieses Projekt war nicht unser Favorit, ja es war nicht mal auf unserer Wunschliste. Es war ein Projekt, welches auf einer Studienarbeit aufbaute mit Technologien, die wir noch nie verwendet hatten. All diese Punkte wirkten anfangs eher abschreckend. Zum Glück hatten wir ein Betreuer-Team, welches uns sehr gut unterstützte und uns einen guten Einstieg in das Thema ermöglichte. Spannende Theorieblöcke zu Computernetzen und Segment Routing trieben die Arbeit und den Wissensstand voran. Immer wenn Fragen aufkamen, konnten wir uns an einen Betreuer wenden, welcher sich die Zeit nahm, diese zu beantworten. Im Allgemeinen verlief die Zusammenarbeit mit den Parteien reibungslos und es herrschte eine angenehme Atmosphäre.

In den ersten Wochen wurden wir damit beauftragt, uns in die Technologien Kafka und Spring Boot einzuarbeiten, sowie die vorhergehende Studienarbeit zu lesen und den bestehenden Code zu analysieren und gegebenenfalls zu überarbeiten. Bis zu diesem Zeitpunkt musste ich in meinem Studium noch nie an bestehendem Code weiterentwickeln. Dies hat mir einen guten Einblick gewährt, was mich in der Praxis mit grosser Wahrscheinlichkeit erwarten wird und wie wichtig sauberer Code ist. Einfache Refactorings hätten uns möglicherweise geholfen, uns schneller im Code zurechtzufinden.

Das Zusammenspiel von Docker, Kafka und Spring Boot in einer Netzwerk Domäne war hochspannend. Zugegebenermassen war es anfangs ein steiniger Pfad. Vor allem das Beheben des Path-Finding-Bugs kostete viel Zeit und Nerven. Umso schöner war dafür der Moment, als der Fehler gefunden und die Pfade richtig dargestellt werden konnten! Man könnte schon fast behaupten, dass der Pfad das Ziel war. Und auf diesem Pfad konnte ich umfassendes Fachwissen erlangen und hatte grossen Spass dabei.

D.2 Benjamin Moosmann

Meine Bachelorarbeit habe ich mir reichlich anders vorgestellt. Ich bin davon ausgegangen mit meinem gewohnten Stack von Entwicklungs-Tools eine vielleicht etwas kompliziertere CRUD Applikation auf einer "grünen Wiese" zu erstellen.

Stattdessen haben wir eine Arbeit zugewiesen bekommen, die alles andere ist. Die vorgegebenen Technologien waren zu Beginn her alle neu, es bestand bereits ein Prototype einer Semesterarbeit und ein detailliertes Wissen der abgefragten Netzwerk Technologien war unumgänglich. Trotz dieser anfangs etwas unangenehmer Ausgangslage haben wir uns nicht unterkriegen lassen und uns vertieft in die neuen Technologien und Konzepte eingearbeitet. Docker im Zusammenspiel mit dem «.gitlab-ci.yml» hat sich dabei schnell als unsere neuer Liebling für das Deployment herauskristallisiert. Auch die Umstellung auf Latex mit Overleaf hat sich als grossen Gewinn herausgestellt. Die Einarbeitung in das Java Spring Boot Backend ging ebenfalls relativ gut. In das Frontend mit der Vue.js App hatte vor allem Patrick viel Zeit investiert und mich immer tatkräftig unterstütz. Was sich hingegen als harter Brocken herausgestellt hat, ist die Verwendung von Kafka. Das einfache «producen» und «consumen» von Stream Daten war schnell verstanden, aber das Zusammenspiel von unterschiedlichen und zeitlichen «windowed» Topic erwies sich als äusserst kompliziert. Durch die Kopplung an die Telemetry Devices für den Stream war das Entwickeln und Testen aufwendig und die Resultate teilweise schwierig zu interpretieren.

Die grösste Herausforderung war das Verständnis für die YANG Models. Hier wurden wir vom INS tatkräftig unterschützt, aber selbst gemeinsam hat es mehrere Anläufe und sehr viel Analyse Arbeit benötigt um den Label Stack für einen Pfad richtig mit einem Service (VRF) mappen zu können. Hier haben wir aus meiner Sicht viel Zeit in der Arbeit verloren, zulasten weiterer Features.

Wenn ich die Arbeit noch einmal von vorne beginnen dürfte, würde ich nicht auf dem bestehenden Prototype der Semesterarbeit aufsetzten, sondern diese nur als Inspiration für die fachlichen Lösungen verwenden. Dies würde uns erlauben die einzelnen Komponenten besser voneinander zu trennen und sauberer Softwarecode zu produzieren.

Trotzdem sehe ich die Bachelorarbeit im Gesamten als einen Gewinn. Uns ist es gelungen in kurzer Zeit fundiertes Wissen in vielen unterschiedlichen Technologien aufzubauen und die Core-Anforderungen der Aufgabenstellung in einer uns fremden Domäne umzusetzen.

Appendix E

Time Report

The time report data is exported from Jira stories. Those times got categorized and visualized in diagrams with Excel.

E.1 Time Report by Category

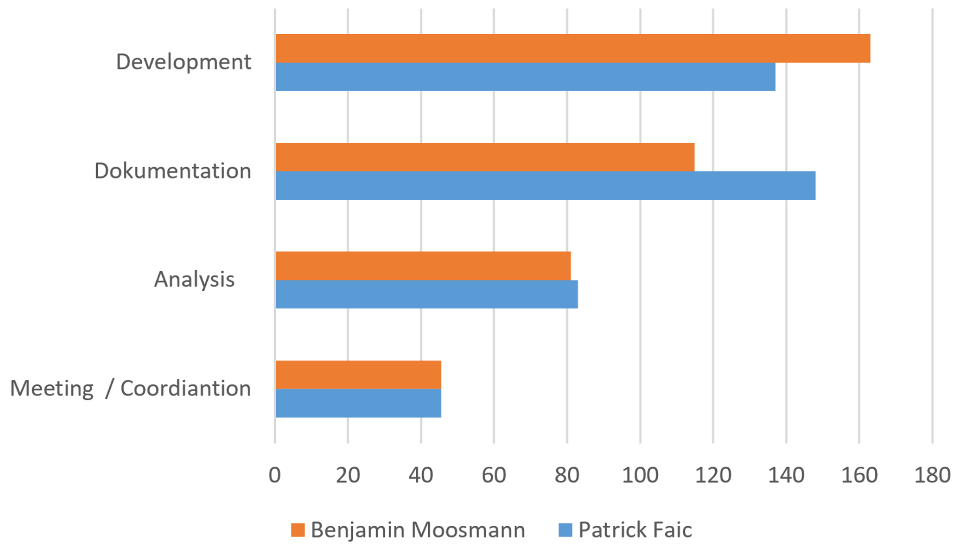


Figure E.1: Time by Category

E.2 Time Report by Team Member

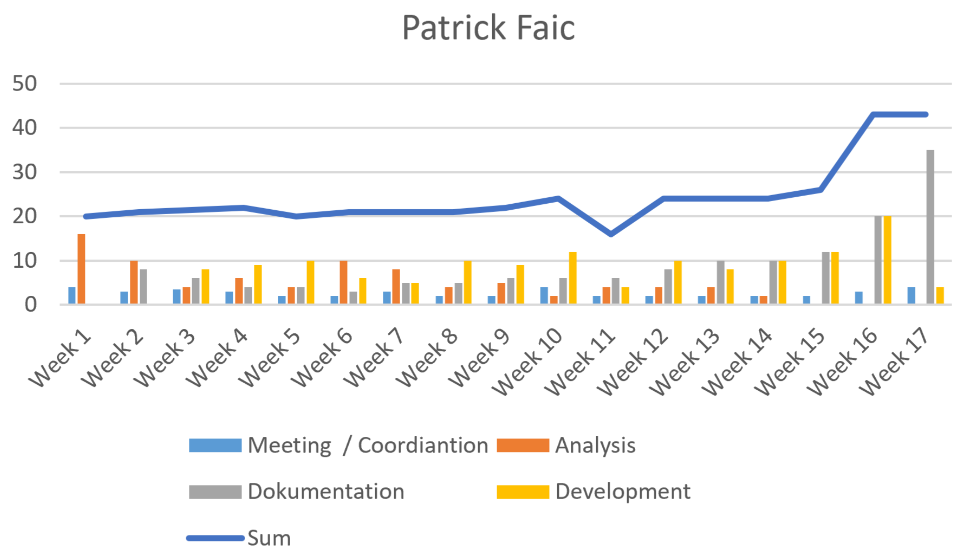


Figure E.2: Time by Patrick Faic

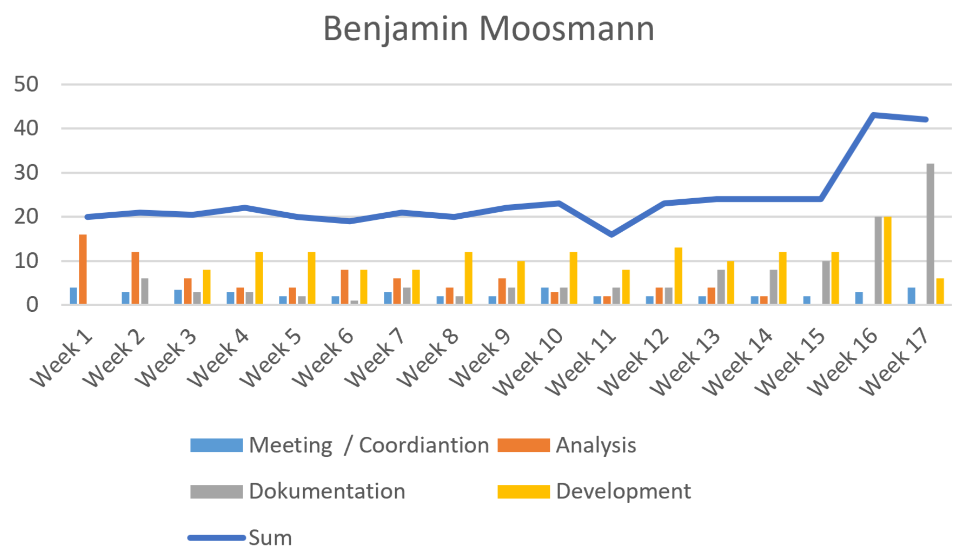


Figure E.3: Time by Benjamin Moosmann

Appendix F

Meeting Protocols

F.1 Meeting Protokoll Woche 1

Ort	8.267
Datum	Mi 20.02.2019
Uhrzeit	9:00 - 10:00
Teilnehmer	• Faic Patrick • Metzger Laurent • Moosmann Benjamin • Schmid Philip • Witmer Marcel (über WebEx)

Aktuelles

1. Vision:

   a) OSPF Routing Protocol. Segment Routing basierend auf einer labled Routing Table.

   b) Shortest Path-Algorithmus verbesseren. Z.B Ein anderer Shortest Path für VoIP als für anderen Datentransfer.

   c) Streaming telemetry mit z.B. GRPC oder JSON. Neu mit dem Push- anstelle des Pull-Prinzip mit SNMP.

2. Bestehendes Konzept:
   Daten werden vom Lab über Cisco Big Muddy Pipelines geschickt und

mit Hilfe von Apache Kafka in einer Webanwendung (Vue.js, d3.js) als Graph dargestellt.

3. Backend Spring Boot benötigt Refactoring, falls das selbe verwendet wird.
4. CE Nodes gibt es nicht mehr.

Beschlüsse

1. Technischer Bericht in Englisch verfassen.
2. Kommunikationsplattform: WebEx.
3. 24h vor Meeting eine Übersicht versenden.
4. Cisco Big Muddy Pipelines verwenden.

Ausblick

1. In die zu Grunde liegende Studienarbeit einlesen.
2. Zu verwendende Tools definieren.
3. Umgebung aufsetzen.

Nächster Termin

Termin	27.02.2019
Bemerkungen	-

F.2 Meeting Protokoll Woche 2

Ort	8.267
Datum	27.02.19
Uhrzeit	09:00
Teilnehmer	• Faic Patrick • Moosmann Benjamin • Schmid Philip • Witmer Marcel

Rückblick

1. Inhaltsverzeichnis analysiert.

Aktuelles

1. Definierte UC besprochen.

Beschlüsse

1. Philip lässt den Studenten SA-Graph-Framework/Library-Evaluation zukommen.
2. Philip schickt den Studenten CloudInf Docker Folien.
3. Spring-Termin mit Matthias (Mittwoch 06.03.2019 14:00 - 15:00)

Ausblick

1. Backend Refactoring.
2. Evaluation von Frontend Frameworks.
3. Projektplan erstellen.
4. In Docker einarbeiten.

Bemerkungen

Nächster Termin	06.03.2019
Abwesend	Laurent Metzger

F.3 Meeting Protokoll Woche 3

Ort	8.267
Datum	06.03.2019
Uhrzeit	9:00
Teilnehmer	• Faic Patrick • Metzger Laurent • Moosmann Benjamin • Schmid Philip

Rückblick

1. Projektplan erweitert (Risikoanalyse, Meilensteine definiert, Stakeholder-Analyse)
2. Graph libraries analysiert.

Aktuelles -

Beschlüsse

1. Zwischenpräsentation an einem Mittwoch vor Abgabe.
2. Prof. Metzger teilt Angaben zu Projektbeteiligten mit.
3. Meilensteine visualisieren.
4. Organisationsstruktur visualisieren.
5. Im Verlauf des Projekts Scalbility dokumentieren.
6. Studenten teilen Doku mit Betreuern.

Ausblick

1. Spring-Meeting mit Matthias.
2. Backend Refactoring.
3. Bereits mit Construction anfangen.

Bemerkungen

Nächster Termin	13.03.2019

F.4 Meeting Protokoll Woche 4

Ort	8.267
Datum	13.03.2019
Uhrzeit	9:00
Teilnehmer	• Faic Patrick • Metzger Laurent • Moosmann Benjamin • Schmid Philip • Witmer Marcel (über Webex)

Rückblick

1. Verbesserungen der Visualisierungen angeschaut.
2. Workflow

Aktuelles

1. Backend-Refactoring dauert länger als erwartet.
2. Gegenleser: Stefan Richter
3. Experte: Laurent Billas (Armasuisse)
4. Kurze Einführung in Watchtower von Schmid.

Beschlüsse

1. 25.04, 13 Uhr Zwischenpräsentation.
2. Kompatibilität mit Microsoft Edge nicht zwingend.
3. Laurent überprüft config von XR-01.

Ausblick

1. Refactoring beenden.
2. Erste Use Cases erledigen.

Bemerkungen

Nächster Termin	20.03.2019

F.5 Meeting Protokoll Woche 5

Ort	8.267
Datum	20.03.2019
Uhrzeit	9:00
Teilnehmer	• Faic Patrick • Metzger Laurent • Moosmann Benjamin • Schmid Philip • Witmer Marcel

Rückblick

1. Watchtower konfiguriert.
2. Jira und GitLab miteinander verbunden.
3. Frontend Refactored -> Framework Library gewechselt.

Aktuelles

1. Routing anhand von Label Stack angeschaut.

Beschlüsse

1. Telemetriedaten aus Big Muddy Pipeline in opt/data Textdatei dumpen und an Metzger senden.
2. Studenten überprüfen XR-01 Daten.
3. Logischer Label-Pfad ist unidirektional und können im UI mit Pfeil dargestellt werden.
4. Mehrere Links zwischen Nodes sollten im UI dargestellt werden können.
5. Beim hovern über einer Node sollen Informationen dargestellt werden.
6. Wie soll physische/logische Granularität dargestellt werden?

Ausblick

1. Beim nächsten Meeting Yang besprechen.
2. UC-7, UC-6
3. Kafka verstehen mit Hilfe von Philip/Matthias.

Bemerkungen

Nächster Termin	27.03.2018
Besonderes	Marcel hat Gipfeli gebracht.

F.6 Meeting Protokoll Woche 6

Ort	8.267
Datum	27.03.2019
Uhrzeit	9:00
Teilnehmer	• Faic Patrick • Metzger Laurent • Moosmann Benjamin • Witmer Marcel (über Webex)

Rückblick

1. Use Case Diagramm
2. UC-8

Aktuelles

1. XR-01 ist in telemetry-lldp-grouped sichtbar, aber nicht in telemetry-nodes

Beschlüsse

1. Dump Files senden.

Ausblick

1. UC-7
2. YANG-Modelle anschauen.

Bemerkungen

Nächster Termin	03.04.2018
Abwesend: Schmid Philip	

F.7 Meeting Protokoll Woche 7

Ort	8.267
Datum	03.04.2019
Uhrzeit	9:00
Teilnehmer	• Faic Patrick • Metzger Laurent • Moosmann Benjamin

Rückblick

1. Dump File analysieren.

Aktuelles

1. Nicht alle virtuelle Maschinen sind aktiv.

Beschlüsse

1. Dumps erstellen fürs Montags-Meeting.
2. Studenten dürfen eigene Ports freischalten auf 10.20.0.10

Ausblick

1. Am Meeting am Montag wird Prof. Metzger den Studenten einen Theorieeinschub zum Thema Segment Routing geben.

Bemerkungen

Nächster Termin	08.04.2019, 11:00 - 13:00 Uhr, Zimmer 8.225
Abwesend: Schmid Philip, Witmer Marcel	

F.8 Meeting Protokoll Woche 8

Ort	8.225
Datum	08.04.2019
Uhrzeit	11:00
Teilnehmer	• Faic Patrick • Metzger Laurent • Moosmann Benjamin

Rückblick

1. -

Aktuelles

1. Analyse der Pfadfindung mit der Hilfe von MPLS, Path Policy und LLDP.
2. LLDP hat Probleme, konstant Nachbar-Nodes zu detektieren. CDP ist in dieser Hinsicht stabiler. Der Aufbau des Protokolls ist der gleiche.

Beschlüsse

1. CDP anstatt LLDP verwenden.
2. XTC-Nodes nicht im Frontend darstellen.

Ausblick

1. Path darstellen.

Bemerkungen

Nächster Termin	17.04.2019

F.9 Meeting Protokoll Woche 9

Ort	8.267
Datum	17.04.2019
Uhrzeit	9:00
Teilnehmer	• Faic Patrick • Moosmann Benjamin • Schmid Philip • Witmer Marcel (Webex)

Rückblick

1. A

Aktuelles

1. CDP Informationen wurden nicht weitergeleitet weil der Pfad nicht auf Containerebene angegeben wurde.

Beschlüsse

1. Philip ermöglicht CDP auf allen Routern.

Ausblick

1. Auf CDP umstellen
2. Valide Pfade im Frontend anzeigen solange es geht.
3. Alle vorhandenen VRFs anzeigen/Fehler finden.

Bemerkungen

Nächster Termin	24.03.2018
Abwesend	Metzger Laurent

F.10 Meeting Protokoll Woche 10

Ort	8.267
Datum	24.04.2019
Uhrzeit	9:00
Teilnehmer	• Faic Patrick • Metzger Laurent • Moosmann Benjamin • Schmid Philip • Witmer Marcel

Rückblick

1. Path dargestellt (Bug fixes: MPLS, Collection-ID filtering)
2. Zwischenpräsentation vorbereitet
3. Path-Finding-Diagramm erstellt.
4. Von LLDP auf CDP
5. XTC-Nodes aus Frontend entfernt.
6. Topologie Darstellung angepasst.
7. Backup-Routen aus Config entfernt. (Prof. Metzger)
8. Encoding-paths bearbeitet. (Philip)

Aktuelles

1. Es sind nur zwei Policy Labels von Router XR-09 vorhanden.

Beschlüsse

1. C

Ausblick

1. Zwischenpräsentation
2. Policy Labels Aggregierung fixen. (Wir sehen nur 2 Policy Labels)
3. Weiteres Vorgehen definieren. (Nächster Use-Case)

Bemerkungen

Nächster Termin (Präsentation)	25.04.2019
Nächstes Meeting	01.05.2019

F.11 Meeting Protokoll Woche 11

Ausfall wegen zu geringem Fortschritt.

Gründe:

- Vorstellungsgesprächen
- Gesundheitliche Probleme
- Abgaben in anderen Modulen

F.12 Meeting Protokoll Woche 12

Ort	8.267
Datum	
Uhrzeit	9:00
Teilnehmer	• Faic Patrick • Moosmann Benjamin • Schmid Philip • Witmer Marcel (über Webex)

Rückblick

1. Dokumentation aktualisiert.
2. Segment Routing Überblicksdiagramm.
3. Probleme mit Policy Label Aggregation.
4. Häufiger Absturz des Docker Containers.

Aktuelles

1. Toolchain mit Java Versionen sauber definieren.
2. Volume von Nodes voll gelaufen.
3. Kafka Workers lokal laufen lassen.

Beschlüsse

1. Restart Policy für Watchtower definieren.
2. Topics löschen.
3. Matthias fragen wegen Kafka Joins.

Ausblick

1. Policy Label aggregieren.

Bemerkungen

Nächster Termin	15.05.2019
Abwesend	Metzger Laurent

F.13 Meeting Protokoll Woche 13

Ort	8.267
Datum	15.05.2019
Uhrzeit	15:00
Teilnehmer	• Faic Patrick • Metzger Laurent • Moosmann Benjamin

Rückblick

1. Probleme zur Findung der Policy Labels dargelegt.

Aktuelles

1. "pushed-mpls-label-stack" = BGP label
2. Es werden die falschen Label zur Aggregation der Segment Routing Informationen verwendet.
3. BGP-Label werden über eine Zwischentabelle auf die Policy-Label gemapped. Dies führt dazu, dass die falschen Keys (<node>-<label>) der Kafka-Topics verglichen werden.

Beschlüsse

1. Neue Sensor Paths auf Node XR-09 konfiguriert: Cisco-IOS-XR-ipv4-bgp-oper:session, Cisco-IOS-XR-mpls-vpn-oper
2. Aktuellen Dump für freitags Meeting erstellen.

Ausblick

1. Neue Sensor Paths auf Vorkommnisse der Labels 24011 bzw. 24012 für XR-09 untersuchen?

Bemerkungen

Nächster Termin	17.05.2019
Abwesend	Schmid Philip, Witmer Marcel

F.14 Meeting Protokoll Woche 14

Ort	8.267
Datum	17.5.2019
Uhrzeit	11:00
Teilnehmer	• Faic Patrick • Metzger Laurent • Moosmann Benjamin • Schmid Philip

Rückblick -

Aktuelles

1. Die im vorhergehenden Meeting definierten Punkten wurden besprochen und aufgrund dessen neue Sensor Paths konfiguriert.

Beschlüsse

1. Neu-konfigurierte Sensor Paths analysieren.

Ausblick

1. Konfigurierte Sensor Paths in Code einbinden.

Bemerkungen

Nächster Termin	29.05.2019
Nächstes Reguläres Meeting fällt aus	

F.15 Meeting Protokoll Woche 15

Ort	8.267
Datum	29.05.2019
Uhrzeit	9:00
Teilnehmer	• Faic Patrick • Moosmann Benjamin • Schmid Philip

Rückblick

1. Policy Label/Path Finding gefixt

Aktuelles

1. Es wurde besprochen, welche Use Cases zeitlich noch erledigt werden können.

Beschlüsse

1. UC-06 wird implementiert.
2. UC-07 wird nur schriftlich beschrieben.
3. Optionale Use Cases werden verworfen.

Ausblick

1. UC-06 implementieren.
2. Management Summary
3. Poster
4. Abstract

Bemerkungen

Nächster Termin	01.03.2018
Abwesend	Metzger Laurent, Witmer Marcel

F.16 Meeting Protokoll Woche 16

Ort	8.267
Datum	05.06.2019
Uhrzeit	9:00
Teilnehmer	• Faic Patrick • Moosmann Benjamin • Schmid Philip

Rückblick

1. Repo Readmes erstellt
2. XR-09 Path bug behoben
3. Datenaggregation verbessert
4. Management Summary geschrieben

Aktuelles

1. Löschen von Kafka-Daten funktioniert nicht. Dies verlangsamt das verarbeiten der Daten und die Nodes werden darum verspätet angezeigt.

Beschlüsse

1. Kafka-Daten wurden im Plenum gelöscht.

Ausblick

1. Abstract (bis am Freitag)
2. Frontend-Feinschliff
3. Poster (Abgabe 12.06 12:00)
4. Doku (API, Scalability)
5. Docker Environment variables

Bemerkungen

Letzter Termin	11.06.2019
Abwesend	Metzger Laurent, Witmer Marcel

F.17 Meeting Protokoll Woche 17

Ort	8.267
Datum	11.06.2019
Uhrzeit	13:00
Teilnehmer	• Faic Patrick • Schmid Philip

Rückblick

1. Readmes besprechen

Aktuelles

1. Muss die originale Aufgabenstellung an den Anfang der Dokumentation gesetzt werden?

Beschlüsse

1. Aufgabenstellung an den Anfang setzen.

Ausblick

1. Doku abschliessen
2. Urheber- Nutzungsrecht unterschreiben lassen
3. Eigenständigkeitserklärung unterschreiben lassen
4. Aufgabenstellung unterschreiben lassen
5. Poster abgeben für Ausdruck
6. Dokumentation abgeben

Bemerkungen

Letztes Meeting	
Abwesend	Metzger Laurent, Witmer Marcel

Glossary and Abbreviations Index

API Application Programming Interface. 104, *see* Application Programming Interface

CDP Cisco Discovery Protocol. 104, *see* Cisco Discovery Protocol

JavaScript Object Notation File format to transmitt data objects. 6

JSON JavaScript Object Notation. 6, 104, *see* JavaScript Object Notation

LLDP Link Layer Discovery Protocol. 104, *see* Link Layer Discovery Protocol

Topic A topic is a concept in Appache Kafka where data streams are published into. Think of a glass that gets filled with water from a faucet.. 30

VRF Virtual routing and forwarding. 104, *see* Virtual routing and forwarding

Literature Index

[1] Inc. Cisco Systems. *Segment Routing*. URL: `https://www.segment-routing.net/` (visited on 06/02/2019).

List of Figures

List of Tables