NTNU
Norwegian University of
Science and Technology

Modelling the dependability in Network Function Virtualisation

Wenqi Lin

Master of Telematics - Communication Networks and Networked Services
Submission date: June 2017
Supervisor: Bjarne Emil Helvik, IIK
Co-supervisor: Gianfranco Nencioni, IIK

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

NTNU – Trondheim
Norwegian University of
Science and Technology

Modelling the dependability in Network Function Virtualization

Wenqi Lin

Submission date: June 2017
Responsible professor: Bjarne Emil Helvik, IIK
Supervisor: Gianfranco Nencioni, IIK

Norwegian University of Science and Technology
Department of Telematics

Title: Modelling the dependability in Network Function Virtualization

Student: Wenqi Lin

Problem description:

- **Background**
  Current service provisioning in telecommunication industry has long product cycles, low service agility and heavy dependence on specific hardware. While at the same time, the customers' demands of having more diverse and new services with high data rates keep increasing. Thus, telecommunication service providers (TSP) intend to meet these high customer requirements, and reduce the high capital expenditure (CAPEX) and operation expense (OPEX). Under these circumstances, Network Function Virtualization (NFV) has been proposed to address these challenges by leveraging virtualization technology to offer TSPs a new way to build more a dynamic and service-aware network. The main concept of NFV is the decomposing the services into a set of Virtual Network Functions (VNF) which are implemented in software running on commodity servers. The NFV management and orchestration (MANO) provides the functionality required for the provisioning of VNFs and, therefore, focuses on all virtualization-specific management task and the lifecycle management of VNFs. Another concept which is closely related to NFV is Software Defined Networking (SDN). SDN decouples the network control and forwarding functions which centralizes the controller and simplifies the network management. Notably, NFV and SDN have a lot in common and are highly complementary since they all use automation and virtualization technology to achieve their goals respectively. However, they are not depending on each other and have been brought up to address different aspects. Even though the NFV has drawn significant attentions from both academia and industry, the development is still at an early stage. Therefore, efforts should be made on addressing various unexplored research challenges such as dependability issues.
- **Objective**
  In this project, the major concern is about the dependability of NFV network services. The focus is to develop a dependability model of NFV that also consider connectivity requirements.
- **Technical approach**
  The work is planned to include the following activities:
  - The functionalities and characteristic of NFV and NFV MANO shall be studied already.
  - The state of the art on the most relevant work on dependability of NFV shall be briefly overviewed and summarized. In NFV, there are many

different possible sources of failures and some of these are depending on the specific provided services, such as the virtualized Evolved Packet Core (vEPC).

- After an accurate evaluation, the potential failures, such as MANO failures, logic connection failures, in the selected scenario shall be categorized.
- A two-level model shall be developed based on the information. In the first level, the structure of the NFV along with the connectivity among different NFV elements in the network will be developed. The second level which is used to evaluate the dependability of the different NFV elements such as the VNF, the MANO and the NFV infrastructure.
- With the help of the model, analysis of evaluating how the different NFV elements influencing the dependability of the network services shall be conducted. Möbius software tool is one of the developing tool which will be used in the project.
- If time allows, the model might be extended to evaluate NFV over SDN architecture.

- **Expected results:**

  - A dependability model of NFV addressing the NFV provisioning and the relevant connectivity required for a working system. (If time allows, the model may also include elements related to the dependability of NFV over SDN.)
  - One or more case studies of a simple or moderately complex network/system.

Responsible professor: Bjarne Emil Helvik, IIK
Supervisor: Gianfranco Nencioni, IIK

Abstract

Network Function Virtualization has been brought up to allow the TSPs to have more possibilities and flexibilities to provision services with better load optimizing, energy utilizing and dynamic scaling. Network functions will be decoupled from the underlying dedicated hardware into software instances that run on commercial off-the-shelf servers. However, the development is still at an early stage and the dependability concerns raise by the virtualization of the network functions are touched on only briefly. Particularly, the evaluation of the NFV-based services' dependability has never been conducted.

Towards this goal, this thesis aims to address the dependability concern of NFV and uses a two-level availability approach to construct a quantitative evaluation about how to assess the availability of an NFV-based service and how NFV elements in the network shall be deployed to provide a more dependable network service.

A two-level availability model has been developed based on the various NFV-based network. The first level is focusing on the topology of the network and the connectivity requirements for provisioning an NFV-based service. In the second level, the Stochastic Activity Network (SAN) model of different network elements such as VNF, NFV-MANO and datacenter have been developed to evaluate the availability. Eventually, these two types of models have been merged together to illustrate the overall availability/unavailability of the NFV-based services in different use cases.

In the end, analysis and evaluation have been conducted based on the obtained results from the two-level availability model. There are seven different scenarios have been simulated with regard to the deployment of NFV across the network. And the outcome on how the variations of the NFV elements deployment influence the dependability of the NFV-based services will be presented along with some suggestions about the NFV deployment in provisioning an end-to-end service.

Preface

The thesis is presented as an accomplishment for the Master of Science degree at the Department of Telematics in Norwegian University of Science and Technology (NTNU).

Firstly, I wish to express my sincere gratitude to my supervisor Gianfranco, a postdoctoral researcher in NTNU for his support, guidance and suggestion during these 21 weeks. He always gives me very constructive suggestions and feedbacks that have enabled me to complete my thesis.

Secondly, I sincerely thank my responsible professor Bjarne Emil Helvik for his guidance and encouragement in carrying out this project work. I have learned a lot from him in the past two years. He gives me valuable feedbacks not only to this thesis work but also to my profession skills.

Furthermore, it is my privilege to thank my boyfriend Yuting Situ, for his constant encouragement and caring in the research period.

Finally, I am truly grateful and thankful to my family in China, they are very supportive, either morally, financially and physically. I could not have done this without your help.

Contents

List of Figures

List of Tables

List of Acronyms

BSS	Business support systems.
CAPEX	Capital Expenditures.
COTS	Commercial Off-The-Shelf.
CPE	Customer premises equipment.
EM	Element Management.
EPC	Evolved Packet Core.
ETSI	European Telecommunication Standards Institute.
FCAPS	Fault, configuration, accounting, performance, security.
IP	Internet Protocol.
ISG	Industry Specification Group.
MANO	Management and Orchestration.
NAT	Network Address Translation.
NF	Network Function.
NFV	Network Function Virtualization.
NFVI	Network Function Virtualization Infrastructure.
NFVI-PoP	NFVI Point of Presence.
NFVO	NFV Orchestrator.
NIST	National Institute of Standards and Technology.

O&M	Operation and Maintenance.
OPEX	Operating Expenditures.
OSS	Operations support systems.
SDN	Software Defined Network.
TSP	Telecommunication Service Provider.
vEPC	virtualized Evolved Packet Core.
VIM	Virtualised Infrastructure Manager.
VM	Virtual Machine.
VNF	Virtual Network Function.
VNFM	VNF Manager.

Chapter 1
Introduction

This chapter provides a short introduction of the motivation and objectives of the thesis work. A brief overview of the methodology that applied throughout the whole thesis work procedures will be given. In the end, the outline of the thesis will be presented.

1.1 Motivation

The telecommunication service provisioning infrastructure has been developed over decades and has grown to an exceedingly complex system with a growing number of physical proprietary devices and hardware [31]. To deliver an end-to-end service, designated equipment shall be deployed for implementing the individual service functions that make up the service. These service functions shall be chained together in a specific order as a service and provided to the subscribers [29]. Hence, the service provisioning is highly depending on specialized hardware. Moreover, since the services shall be provided as high quality and dependability services, these all contributes to the longer services' life cycles and lower service agility [31].

In the past few years, the Internet has been evolved into a content-based network. Accordingly, the Internet users are getting less satisfied with the existing services and they require more and more different kinds of services at affordable prices [30]. Thus, Telecommunication Service Providers (TSPs) need to adapt themselves to these changes by purchasing more hardware, improving the technical skills, maintaining the equipment and expanding the network. All these requirements will be achieved at high capital expenditures (CAPEX) and operating expenditures (OPEX) for TSPs. However, the TSPs cannot increase the subscription fee accordingly, since the previous experience shows the rise in price only leads to a fierce competition among TSPs and customer might churn. Meeting the customers' growing demands and at the same time maintaining low capital and operating costs become increasingly unrealistic [32]. Therefore, TSPs need an innovative network service paradigm to

cope with the declining profitability and alleviate the problems such as long product cycles, low service agility, and so on [31][29].

Network Function Virtualization (NFV) has been brought up in to take advantage of the virtualization technology to eliminate these problems. The virtualization technology has emerged as a way to reduce the dependency on the dedicated hardware. It decouples the software applications from where they originally belong— the underlying hardware. Then the software applications are able to run in a virtualized environment along with the virtualized hardware resources. NFV is appying the virtualization technology so that the network functions will be decoupled from the underlying dedicated hardware into software instances that run on commercial off-the-shelf (COTS) servers. The main concept of NFV is decomposing the services into a set of Virtual Network Functions (VNFs) which are software instances and can be deployed either from the same geographical location or from the different geographical location. By applying the NFV technology, the TSPs have more possibilities and flexibilities to provision services with better load optimizing, energy utilizing and dynamic scaling [31].

NFV has already drawn immense attentions from researchers in both academia and industry, but the development is still at an early stage. Efforts should be made on addressing various unexplored research challenges and the virtualization of NFs raises performance, dependability and reliability concerns [10]. Dependability issues are touched on only briefly. Particularly, the evaluation of the NFV-based services' dependability has never been conducted. To this purpose, this thesis represents a step towards to the dependability concern of NFV and uses a two-level availability approach to construct a quantitative evaluation on the dependability of the NFV-based services. This thesis answers two quesions: how the availability of an NFV-based service shall be assessed and how NFV-elements in the network shall be deployed to provide a relatively more dependable network service.

This master thesis is regarding the dependability of the network services provisioned in NFV-based networks. The major concerns of the thesis are identified below:

- Literature studies of the difference between current service provisioning method and NFV-based service provisioning method as well as the functionalities and characteristics of NFV.
- Develop a model to evaluate the dependability of NFV-based services by applying a two-level modelling approach. The first level is focusing on the topology of the network and the connectivity requirements for provisioning an NFV-based service. In the second level, the Stochastic Activity Network (SAN) model of different network elements such as VNF, NFV-MANO and datacenter have been developed to evaluate the availability of the selected

network elements. Finally, these two types of models will be merged together to illustrate the overall availability/unavailability of the NFV-based services in different use cases.

– Analyze and evaluate the conducted use cases and present the outcome on how the variations of the NFV elements deployment influence the dependability of the NFV-based services.

1.2 Methodology

The methodology followed in this thesis work is divided into three parts. The first part is the literature study and the second part is to construct a quantitative evaluation. The third part is the analysis. These three steps are shown in the figure below and will be briefly described below in the subsections.

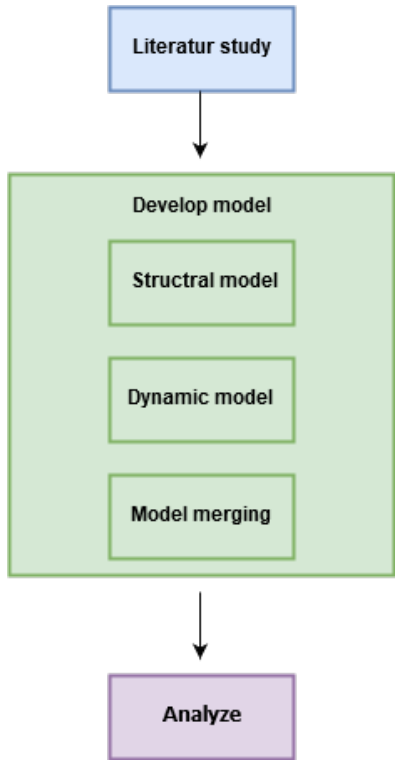

Figure 1.1: Research methodology

1.2.1 Literature study

The literature study is the foundation of the entire thesis work. In the beginning, the basic concepts and related research on NFV and the dependability concerns of NFV have been studied from the existing literature to form a theoretical background of my research area. And then by comparing my research and the existing research, it gives me inspirations and broaden my knowledge during the research methodology development phase. In the later phase, the literature study helps to integrate my findings into the existing body of knowledge [27].

1.2.2 Quantitative evaluation

A two-level availability model will be developed in this phase. This availability model approach is inspired from [17] where the approach is used to address the dependability issues in Software-Defined Networking (SDN).

In the two-level availability model, two different types of model will be developed. The first level is structural model which is focusing on the network structure. The network that deploys the NFV elements will be applied the structural analysis method based on minimal-cut sets by using the Mathematica software tool [41]. The obtained minimal-cut sets then will be analyzed. In the second level, the network elements that composed of the NFV-based network in the first level will be modeled. The possible reasons result in a failure in the network elements will be estimated and modeled by using Möbius software tool [5]. Hence,the overall network service availability can be obtained by merging the two level availability models together by using inclusion-exclusion principle in Mathematica again. The final evaluation and analyzation will be based on the obtained network service availabilities.

1.2.3 Analysis

In the analysis phase, the obtained minimal-cut sets and the overall network service availability will be evaluated to present the variation of the network service availability by deploying different number of NFV elements in the network.

1.3 Outline of the remaining thesis

The thesis is structured as five chapters, and the rest of this thesis is organized as following:

- Chapter 2 provides the background knowledge that learned from the literature studies. The related NFV concepts and deployment ways as well as the dependability issue in NFV will be presented.
- Chapter 3 introduces the two-level availability model approach in detail.Firstly, the structural model development approach will be explained along with a

reference scenario. Secondly, the dynamic model approach will be briefly explained. The SAN model the five selected network elements will be developed and demostrated. In the end, the principle of merging the results from two models will be briefly presented so that the overall network service availability will be achieved.

– Chapter 4 contains the evaluation and analysis of the results in each level.
– Chapter 5 summarizes the paper along with the contributions of the thesis work.

Chapter 2
Background

The chapter presents an overview of the background knowledge as the outcome from the literature studies. It starts with the introduction of the fundamental concepts and architectures of NFV technology. Then the related two technology will be briefly introduced. In addition, the deployment of NFV technology will be viewed and explained as a basis for Chapter 3. At last, another key concept of dependability which will be briefly carried out in the last section along with the related work.

2.1 NFV

As the current telecommunication network service provisioning shown in Figure 2.1, the customer premises equipment (CPE) with several needed service functions have been put at the premises of each subscriber to compose the needed services. For example, if the two service network functions showed below are part of a service chain, the firewall might need to be provided before the Network Address Translation (NAT) to monitor and control the incoming traffic data.

To deliver a network service in such network infrastructure is highly on the dedicated underlying hardware in the CPE, if the TSP has the willing to update the existing service functions or add new service functions, the technicians from the TSP must come to each CPE to do the maintenance work which not only takes time but also costs money. Moreover, the network service delivering highly depends on proprietary hardware appliances. When the TSP attempts to deliver a new service, dedicated hardware must need, this may result in a long service life cycle. To that end, NFV has been proposed by European Telecommunication Standards Institute (ETSI) as a novel approach to accelerate the service delivering. By virtualizing the network functions, the network functions do not rely on the underlying dedicated hardware anymore but instead running on COTS servers as software instances as Figure 2.2 shows.

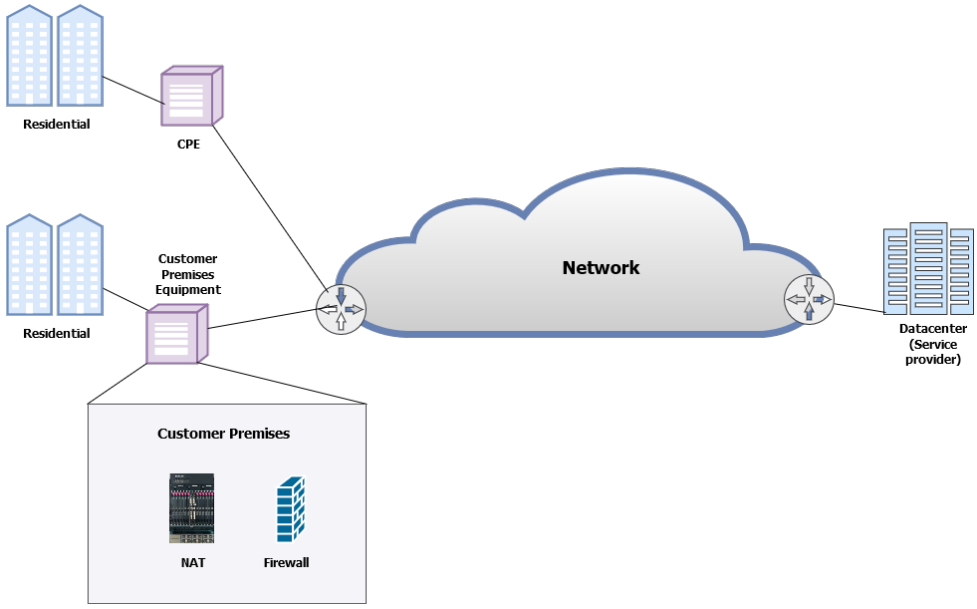

Figure 2.1: Current service provisioning.

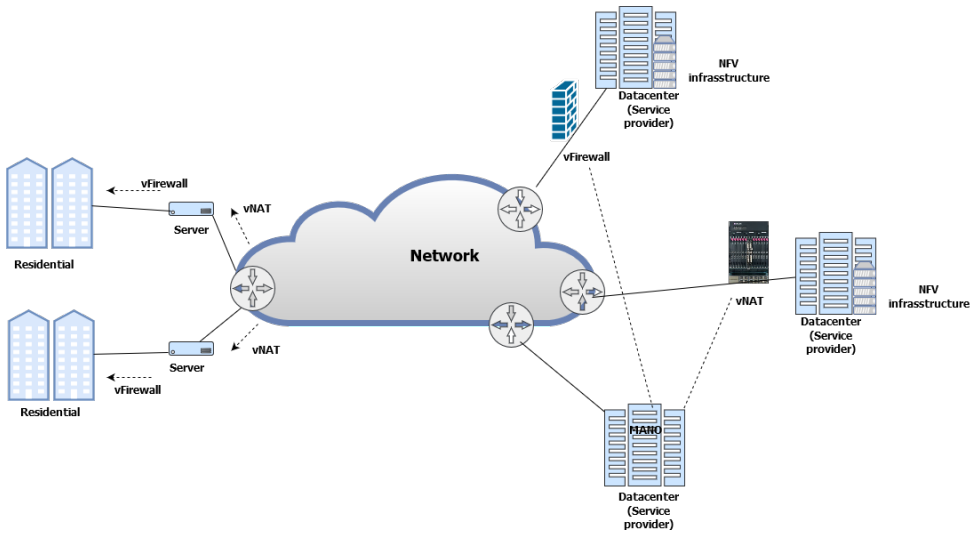

Figure 2.2: NFV-based service provisioning.

Figure 2.2 depicts the NFV-based service provisioning network. The network functions are virtualized as software instances (for example vfirewall, vNAT) sending from the different NFVI-based datacenters. The MANO located in other datacenter is connected to each VNFs to manage and orchestrate the VNFs into a network service function chain. The network service function chain then will be delivered to each subscriber.

ETSI has established the Industry Specification Group (ISG) for NFV aiming to provide a consistent approach to achieve the common architecture that supports VNFs. In October 2012, the telecommunication carriers in the ISG published a white paper that introduces the concept of NFV as well as the benefits, enablers, and challenges this technology. The beginning of NFV starts from then, and the NFV ISG has quickly attracted attentions from a large number of vendors and IT specialists in many different countries. And the participants and members are still increasing as time passes by.Initially, ETSI ISG NFV community published papers about pre-standardization studies. As time goes by, the NFV community has already evolved to the detailed specifications investigation and also the interoperability events. In addition, they are still working on further development of the required standards for NFV and at the same time sharing their experiences of NFV implementation and testing [6].

2.1.1 NFV architecture

ETSI defined three key elements that composed of NFV architecture. The three elements are Network Function Virtualization Infrastructure(NFVI), NFV management and orchestration (MANO) and VNFs which are presented in Figure 2.3.

NFVI
The NFVI contains both software resources and hardware resources to support VNFs. The underlying hardware resources include computing, storage and network resources that provide processing, storage and connectivity to VNFs. The virtualization layer, more specifically, the hypervisor is used to abstract the physical resources into the virtual resources that running in one or multiple Virtual Machines (VM). The virtual resources then will be used by VNFs.

NFV-MANO
From the name of NFV MANO, it can be seen that this element deals with NFV management and orchestration. NFV MANO provides the functionality required for provisioning VNFs. It consists of three functional blocks: Virtualised Infrastructure Manager (VIM), NFV Orchestrator (NFVO) and VNF Manager (VNFM) which are presented in the blue area in Figure2.4.

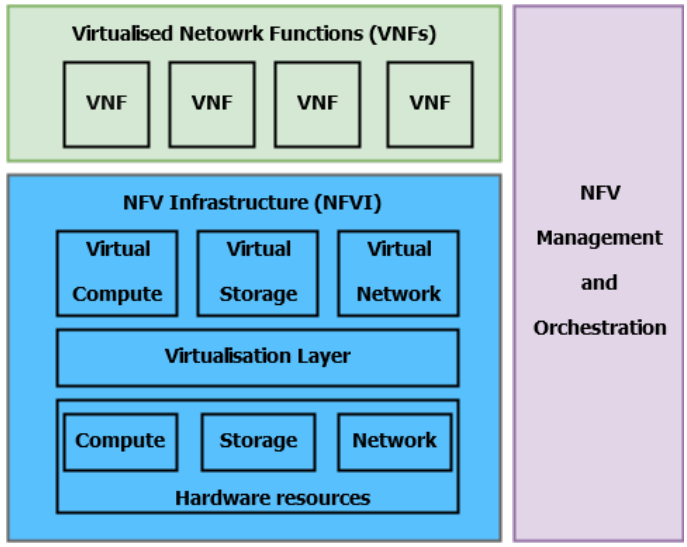

Figure 2.3: High-level architectural framework of NFV [21].

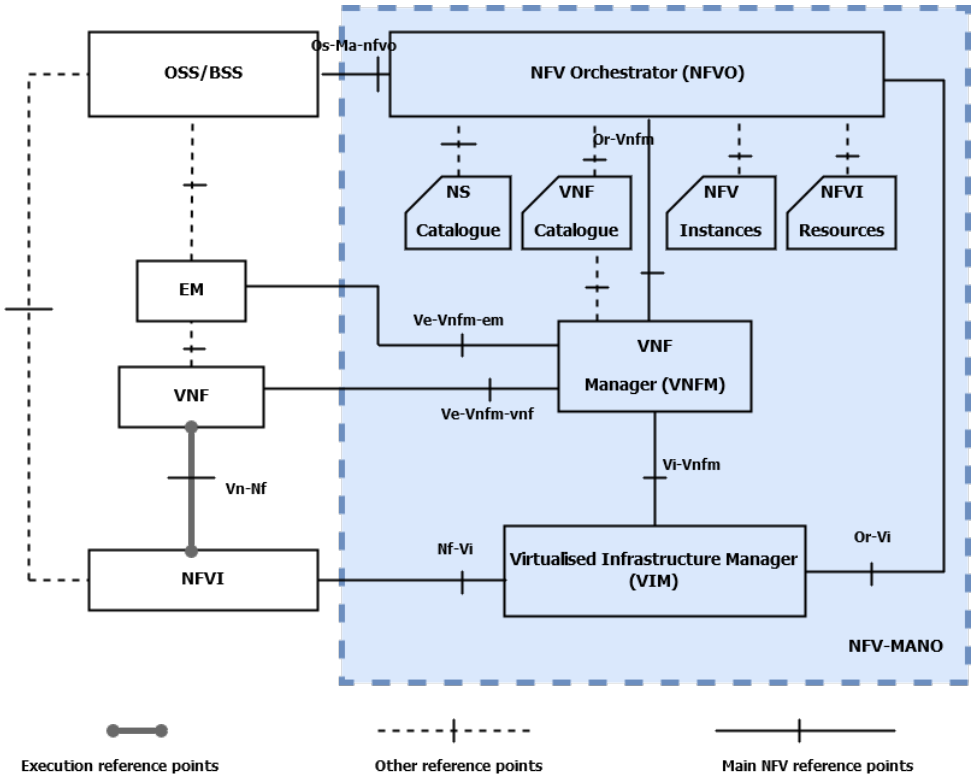

Figure 2.4: The NFV-MANO architectural framework [23].

The VIM is to manage and control both the physical and virtual resources in NFVI through the different reference point interfaces. An NFV architecture may have more than one VIM so that different infrastructure providers are able to manage and control the NFVI resources [33]. Each VNF instance supposes to have an associated VNFM to manage the life cycle of the VNF. However, VNFM does not have the network management to deal with the Fault, configuration, accounting, performance, security (FCAPS) management for the VNF it manages. Therefore, the Element Management (EM) will be used to manage the FCAPS issues as a complement. The NFVO aims to orchestrate different VNFs together into a service function chain so that a given service will be provisioned from a TSP. In addition, NFVO is also able to interact with the existing Operations support systems (OSS)/Business support systems (BSS) [26].

MANO is also a software instance just as VNF, but the difference is MANO do not need to base on an NFVI but rather use the different types of reference points to manage and interact with NFVI and VNF just as Figure 2.2 showed.

VNF
VNFs are software instances that running in one or more VMs. A Network Function (NF) is a function block that resides in the network infrastructures. For example: firewalls, Residential Gateway (RGW), etc [24]. Instead of implementing a NF in a dedicated hardware, a VNF is a just a software instance running in a common commercial-off-the -shelf (COTS). Furthermore, a single VNF might compose of several components, therefore it could be deployed in multiple VMs. However, whether the service is provided through VMs or dedicated physical equipment, the users should not notice the performance difference.

2.1.2 Related concept

NFV is not the only technology that takes advantages of virtualization. There are two other concepts that are similar to NFV and closely related to the virtualization evolvement. The first concept is cloud computing and the second concept is SDN. These three technologies can either work together or work individually.

A. *Cloud computing*
National Institute of Standards and Technology (NIST) in the USA published the definition of cloud computing in [15]. Cloud computing can be seen as a shared pool, the resources such as storages, services and so on are running in the pool and can be accessed conveniently and ubiquitously. Thus, it offers a new mechanism to provision and releases service in a more effective, and at the same time, it reduces the management effort and service provider interaction.

The traditional way to provision a service will be changed from a TSP's prospective. The providers will be divided into two types in this regard. One is infrastructure providers which are responsible for managing and maintaining the cloud platform, at the same time leasing the resources to service providers. The service provider then can rent the resources from either one infrastructure provider or multiple infrastructure providers to deliver the service to the consumers [42].

B. *SDN*
SDN is currently one of the most attracting technologies in both academia and industry. It deals with large-scale complex networks. Originally, these large-scale complex networks may require re-policing or re-configurations from time to time. SDN decouples the network control and the basic forwarding functions. Therefore, SDN makes network control become directly programmable via an open interface. The underlying infrastructure simply do the forward action.

C. *Relationship between NFV and the other two concepts*
These three technologies all can help the telecommunication operators to achieve a scalable, agile and automated network with cost and resource effective. But they address different aspects of the network. Cloud computing is the abstraction of compute resources, SDN is the abstraction of network resources and NFV is the abstraction of function resources.

Cloud computing has already been implemented in most industries and many IT applications have already run on commodity servers in the cloud [39]. And since the performance and reliability requirements of carrier-grade functions are stricter than those of IT applications. If NFV wants to rely on the could computing, many things should be considered and prepared.

SDN and NFV are highly complementary, but they also are two independent technologies. NFV can be used without SDN and vice versa. By applying the SDN concept in NFV, the configuration can be managed in a remote centralized controller to ensure the maintenance and operation procedures become agility and take a shorter time. Meanwhile, it provides a simple way to compatible with the existing deployment [3]. And for SDN applies NFV in the network, the physical hardware resources can be virtualized as software resources. Hence, it reduces the dependence on the dedicated computing hardware which reduces the CAPEX.

2.1.3 NFV Deployment

In the specification [22] published by ETSI, nine different use cases have been proposed that can be implemented by deploying NFV as an attempt to eliminate the existing problems. Based on these use cases, the research community has implemented some of the use cases and try to determine performance characteristics. For example, [35]

is the implementation of Evolved Packet Core (EPC), [38] is the implementation of CPE. There are also a number of equipment vendors deployed NFV from the industry view, such as CloudNFV [2], Huawei NFV Open Lab [4] and etc. Most of the implementations are taking advantage of current SDN and cloud computing technology [31]. As we mentioned in the previous subsection, these three technologies can be used independently, however, using SDN and cloud computing to support NFV can be a better solution.

Among all the NFV implementations, the NFV deployment in our thesis work focuses on a more general service provisioning method. The NFV element shall be deployed either in the same location or distributed in all possible locations. Furthermore, the different VNFs shall be logically chained together so that it could compose of the service that delivered to the end-point. In Figure 2.5 below, there is an example regarding VNF forwarding graph.

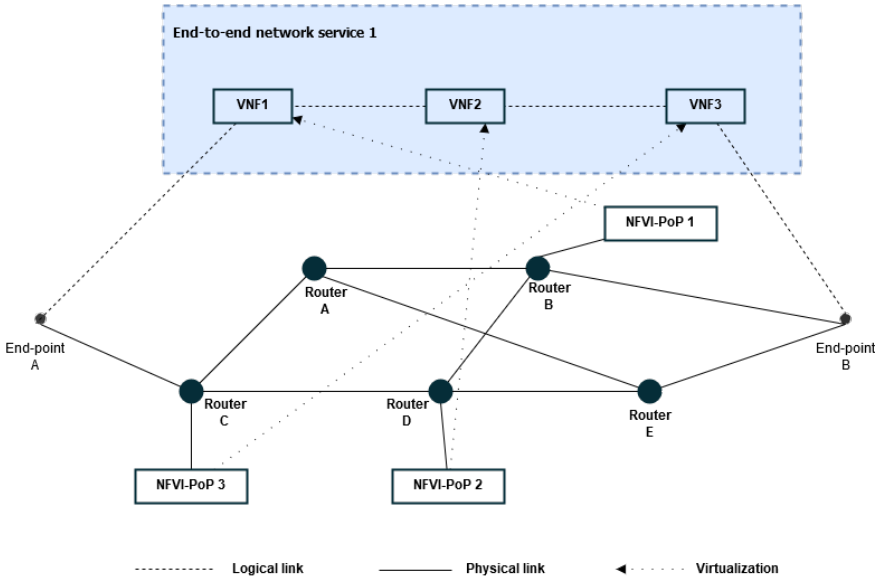

Figure 2.5: Example of end-to-end service chain.

NFVI Point of Presence (NFVI-PoP) represents a place that an NF could be deployed as a VNF [24]. An NFVI-PoP could be a datacenter, a commercial server or other suitable options. In this thesis context, the datacenter and NFVI-PoP will be used interchangeable. The solid lines represent the physical links that connected to different hardware and the dashed lines without an arrow represent the logic links. In addition, the dashed lines with an arrow connect the NFVI-PoPs with the appointed

VNFs meaning the appointed VNF is a software instance running in that NFVI-PoP. The end-to-end network service 1 is provisioned to End-point A from End-point B and it consists of three VNFs which locate in different NFVI-POPs. The sequences of chaining the different VNFs are defined in VNF forwarding graph. And the VNF forwarding graph not only provides a logical connectivity between VNFs but also interconnect with physical NFs to provide a network service to end users [22].

Since the datacenter will be used to support VNFs, the structure of the datacenter will be studied. A datacenter with NFVI consists of three layers presented in Figure 2.6.

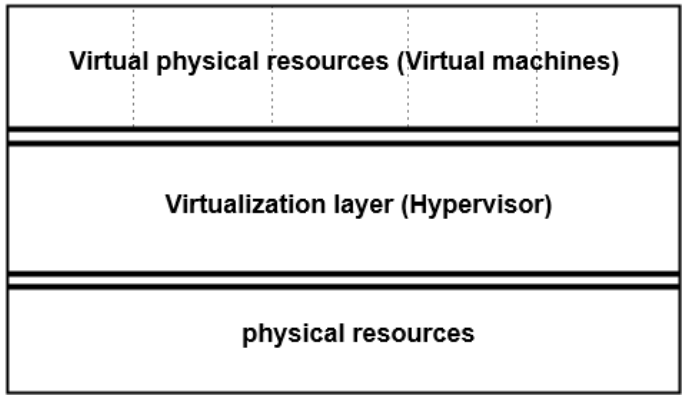

Figure 2.6: Datacenter(NFVI) three-layers model.

The underlying layer is the hardware layer that providing the physical resources, and the middle layer is the virtualization layer which virtualizes the physical resources to be used in the virtual machines by VNF.

There are three types of VNF deployment showing in Figure 2.7 and Figure 2.8. The simplest approach is showed in (a). VNF1 is a software instance that running in a dedicated VM1 along with the virtual resources abstracted by a single hypervisor that locates in a single hardware. Another deployment method is having multiple different VNFs with dedicated VMs that sharing the resources in the same hypervisor and hardware. Example (b) shows VNF1 and VNF2 are sharing the same hardware. That is to say, the physical resources are divided into two sets of resources for VNF1 and VNF2 respectively. An advantage of this approach is that the high utilization of hardware will be achieved [20].

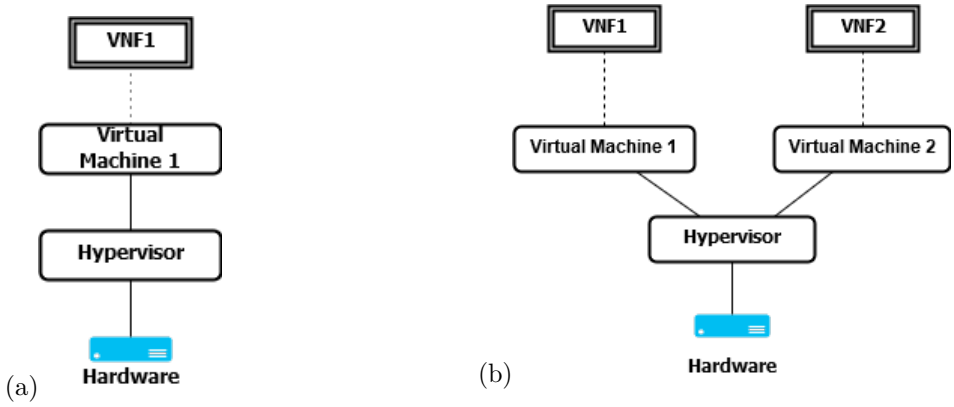

Figure 2.7: Deployment options of VNFs (a) (b).

The most complex VNF deployment is showed in Figure 2.8. VNF1 is composed of three VNF components that executed in dedicated VMs. Two of the VMs are sharing the virtual resources from the same hardware-hardware 1. The other VNF component is executing in another VM1 and using the virtual resources that offered by hardware 2.

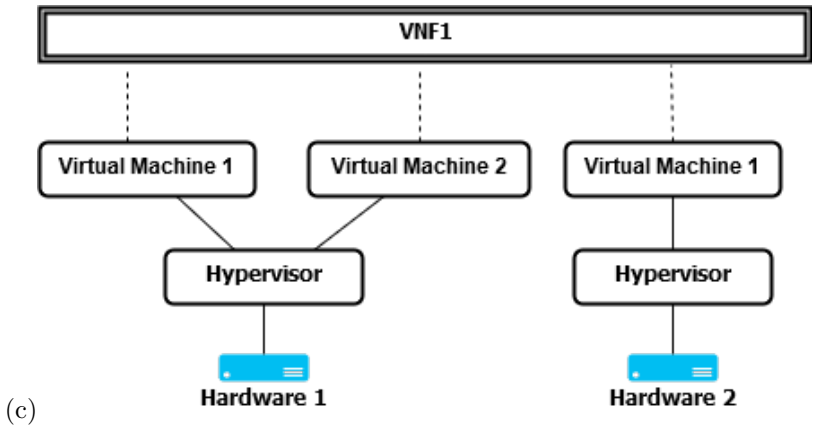

Figure 2.8: Deployment options of VNFs (c).

2.2 Dependability

As the title stated, the focus of this thesis work is related to the dependability concerns of NFV-based networks. Before we take a further step into the thesis, the dependability concept and attributes shall be well explained to gain some insights.

Dependability is referring to the ability of a system that is able to deliver the services to the users [28]. It is a crucial non-functional property of a system, a subsystem, or even an atomic component that forming a unified whole [12]. Faults, errors and failures are the three major threats of system dependability. And to prevent such threats, a dependable system shall equip approaches such as fault prevention, fault tolerance and so on. Depending on the different system or service requirements, different means of avoiding threats will be used in the system.

Dependability has many attributes such as availability, safety, maintainability and so on. These attributes intend to emphasize different facets of system dependability. In reality, there are fewer systems that designed to operate constantly without interruption and other maintenance actions. In many occasions, the number of failures, the probabilities of these failures and the repair time for these failures are significant to measure. To that end, with respect to our thesis work, only availability attribute will be introduced.
Availability: Ability of a system to provide a set of services at a given instant of time or at any instant within a given time interval [12].
Normally, availability is used as a measure of dependability to evaluate systems that short interruptions can be tolerated. For example, when a person watches videos online, he expects to have a good quality and smooth watching experience. And if the video pauses for 100 milliseconds or the quality of the video is not good for 2second, it typically considered acceptable. The time that video supposed to function but not function is considered as downtime. In the table listed below shows the availability and the related downtime per year [11].

Availability (%)	Downtime
99	3.65days/year
99.9	8.76h/year
99.99	52.56min/year
99.999	5.26min/year
99.9999	31.54s/year

Table 2.1: Availability and the related downtime per year.

In telecommunication, “five nines” are often used to represent as a high availability standard. It indicates a system can have downtimes or it can fail at most 5.26 minutes per year. A system that achieves or exceeds this standard often called as carrier grade and it refers to the system is extremely reliable and capable of recovery very fast through redundancy. Usually it takes less than 50 milliseconds to recover [1].

2.3 Related work

Even though NFV has drawn a lot of attention in many, and dependability is becoming a significant issue that makes NFV a success. There is still not much research work that deals with the dependability issue.

In [10], by applying fault injection method to the NFV Infrastructures (NFVI), a dependability evaluation was conducted. The aim is to build confidence in the reliability of NFVIs, to expose the weak points and to provide a practical guidance for designers [10]. This work focuses on the dependability concern in the NFV infrastructure part.

European Telecommunication Standards Institute Industry Specification Group (ISG) NFV published a set of requirements and specifications for designers to help them to develop robust NFV based services in [25]. Moreover, a few techniques and mechanisms are defined in there to ensure reliability and availability in an operational virtual environment [13]. To complement the requirements and specifications in [25], [13] presented the SAN system models to study the sensitivity of the network availability to the main parameters in one of the NFV use cases, vEPC. And in [17], a two-level availability model has been brought up to address the dependability issues in SDN. Only structural vulnerabilities were demonstrated in the paper to be compared in between the traditional Internet Protocol (IP) network and SDN network. Additionally, [34] performed a quantitative assessment to investigate the factors that influence the overall availability in SDN backbone networks. A two-level availability model was developed both in Norwegian national backbone network and a worldwide backbone network. The evaluations were carried out based on the availability models to evaluate which and how the different SDN elements affect the overall network availability.

Chapter 3
Two-level availability model of NFV-based services

In this chapter, the two-level availability modeling approach will be introduced in detail along with an example use case to fully illustrate. The approach will be used to evaluate the dependability of NFV-based services in Norwegian national backbone network.

3.1 Two-level availability model introduction

A two-level hierarchical availability model is used to evaluate the dependability of the network services that provided in the NFV-based networks. It consists of the structural model of the NFV-based network topology and the dynamic model of the network elements. Importantly, this approach is used to model not only a large-scale network but the details of the network as well. The two-level availability modeling methodology has already been considered and used in [34] to measure the dependability in SDN network. In [25], ETSI defines service availability is on a service basis. In other words, service availability in NFV refers to end-to-end service availability which includes all the network elements such as VNFs, NFVI, NFV-MANO. in an end-to-end service. Notably, the endpoints are not included in the measurement [25].

In the structural modeling, the network components in the topology are considered as independent elements. That is to say, a component can be a set of elements that are interdependent and/or experience several failure modes and an advanced recovery strategy [25]. And it applies the structural analysis method based on minimal-cut sets.

In the dynamic modeling, each of the network elements is considered as a single network element with one failure mode. Therefore, Markov model [19] or Stochastic Petri net [16] can be used for modeling. In our case, SAN system model will be used. After developing the two different kinds of models individually, the results

must be merged together to achieve the overall availability of the NFV-based network services.

In the merging phase, the inclusion-exclusion principle shall be used. In addition, the dependability will be measured in terms of steady state availability, hence, the term dependability and availability can be used interchangeably in this thesis context.

3.2 Structural model

Unlike the way that current telecommunication network providing a service to its users, NFV offers a new scheme that the control logic will be moved to centralized in NFV-MANO. And additionally, the different VNFs need to be provided in a chain with a certain order to the user, otherwise, the provided service might not the one that the service provider initially intend to provide. Due to the various added NFV elements, the complexity of the network will increase and the dependability risks will increase accordingly. The focus on the dependability issue will base on providing a service function chain from a service provider node o to a user end-point node d in an NFV-based network.

To provision such a service, the following connectivity requirements must be fulfilled in an NFV-based network:

- **Service function chain.**
  In an NFV-based network, a service will be split into different VNFs which might locate either in different datacenters or in the same datacenters. In the interest of providing the service to the end-users, these functions shall be chained together followed with a specific order defined by TSPs.
- **Connectivity in between at least one NFV-MANO and all endpoints.**
  To provide a service to a user endpoint, the NFV-MANO must connect with the all the endpoints. Hence, the MANO is able to know where to deliver the service and where does the service originate from.
- **Connectivity in between at least one NFV-MANO and all VNF.**
  NFV-MANO is responsible for managing the lifecycle of VNFs and also used to orchestrate the different VNFs together so that a service can be delivered from a service provider endpoint target to a user endpoint. In an NFV-enabled network, VNFs and NFV-MANO might locate in different geographical locations or the same geographical location.

An endpoint is considered as a node or a network that data transmission originates or terminates [40]. The first two requirements are about the necessary NFV elements should connect with all end-points. There is no need to connect all the VNFs to all endpoints, only the first VNF and the last VNF in a service function chain must be connected with the user end-points and the provider end-points respectively. The

last requirement is about the NFV-MANO should be connected to all VNFs, so that it can manage and orchestrate all the VNFs.

To evaluate the dependability of the NFV-based services, the structure-function analysis approach will be applied in Mathematica software tool [41]. By way of explanation, providing a service successfully in a network needs all the components in the network operate successfully or at least one of the minimal-cut set of components are working properly. A cut is a set of components, if all the components in the cut fail, then the system will fail, no matter how other components in other sets behave [9]. Particularly, the minimal set means if remove any component in a set results in the set not to be a cut anymore.

The example shown in below aims at giving a better explanation of the structure-function analysis and minimal-cut set concepts. Specifically, the structural analysis will be applied for the connection from node *n1* to the destination node *n5* in a small-scale network. In the network, there are only 6 nodes and 9 links. And we assume that both the nodes and links might fail.

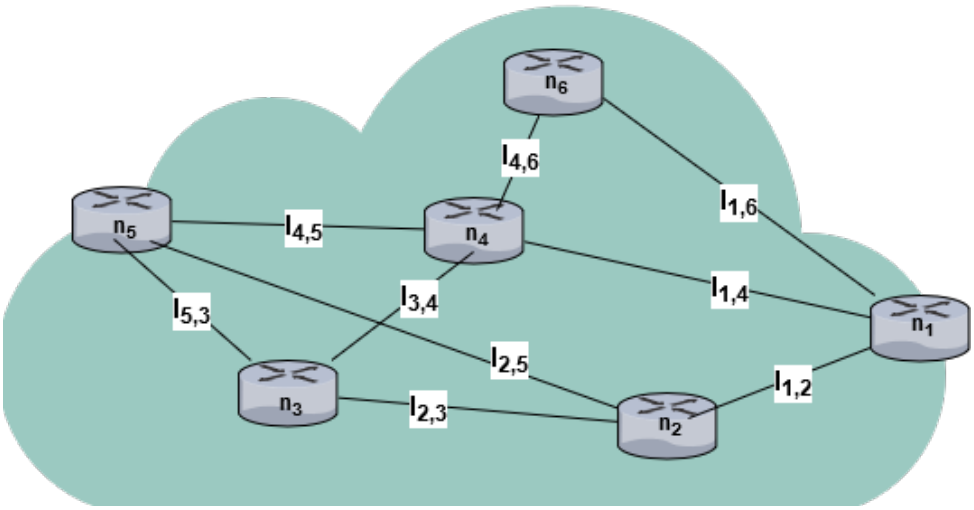

Figure 3.1: Structural analysis of a small-scale network.

By applying structural analysis in Mathematica, the minimal cut sets S will be identified. S is comprised of 31 minimal cut sets.

$$
\begin{aligned}
S = \Big\{ & n_1, \{n_2, n_6, l_{1,4}\}, \{n_2, n_3, l_{4,5}\}, \{n_2, n_4\}, \{n_4, l_{1,2}\}, n_5, \{n_2, l_{1,4}, l_{1,6}\}, \\
& \{n_2, l_{1,4}, l_{4,6}\}, \{n_2, l_{3,4}, l_{4,5}\}, \{n_2, l_{3,5}, l_{4,}5\}, \{n_3, n_4, l_{2,5}\}, \\
& \{n_3, n_6, l_{1,4}, l_{2,5}\}, \{n_3, l_{1,2}, l_{4,5}\}, \{n_3, l_{1,4}, l_{1,6}, l_{2,5}\}, \{n_3, l_{1,4}, l_{2,5}, l_{4,6}\}, \\
& \{n_3, l_{2,5}, l_{4,5}\}, \{n_4, l_{2,3}, l_{2,5}\}, \{n_4, l_{2,5}, l_{3,5}\}, \{n_6, l_{1,2}, l_{1,4}\}, \\
& \{n_6, l_{1,4}, l_{2,3}, l_{2,5}\}, \{n_6, l_{1,4}, l_{2,5}, l_{3,4}, l_{3,5}\}, \{l_{1,2}, l_{1,4}, l_{1,6}\}, \{l_{1,2}, l_{1,4}, l_{4,6}\}, \\
& \{l_{1,2}, l_{2,3}, l_{3,5}, l_{4,5}\}, \{l_{1,2}, l_{3,4}, l_{4,5}\}, \{l_{1,4}, l_{1,6}, l_{2,3}, l_{2,5}\}, \{l_{1,4}, l_{1,6}, l_{2,5}, l_{3,4}, l_{3,5}\}, \\
& \{l_{1,4}, l_{2,3}, l_{2,5}, l_{4,6}\}, \{l_{1,4}, l_{2,5}, l_{3,4}, l_{3,5}, l_{4,6}\}, \{l_{2,3}, l_{2,5}, l_{3,4}, l_{4,5}\}, \{l_{2,5}, l_{3,5}, l_{4,5}\} \Big\}
\end{aligned}
$$

We present the minimal-cut sets into Table 3.1 and use cardinality C_i to categorize the result. When $C_i = k$, the minimal-cut set is composed of a number of sets that only contained k components. For example: if $C_i = 1$, the number of the minimal-cut sets is 2. It implies that there are only two components make up the minimal-cut sets. Either component fails, the whole system will fail as well, even though other components in other sets operate correctly.

	C_1	C_2	C_3	C_4	C_5	Sum
A small-scale network	2	2	16	8	3	31

Table 3.1: The distribution of the cardinality of the minimal-cut sets for a small-scale network.

After equipped the needed background knowledge about the structural analysis and the concept of the minimal-cut sets. An NFV-based network scenario will be conducted as an example and a reference scenario in the later stage. The structural analysis illustrates the vulnerability of an NFV-based network by identifying the minimal-cut sets of the network scenario. The reference scenario is based on the Norwegian backbone network with a single MANO and 3 different VNFs that located in the same datacenter.

The following assumptions shall be considered during the structural analysis:

- Nodes, links, MANO and datacenters in the system may fail;
- The two endpoints and the links between endpoints and nodes will not fail;
- VNF nodes will not fail because they are just abstractions that state the referred VNF;
- The network is working when all the connectivity requirements are fulfilled so that the needed service will be delivered.

Scenario 1 presented in Figure 3.2 contains a three-layer network. The first layer of the network is the national backbone network that consists of 10 nodes. Three

major cities (TRD, BRG, and STV) have duplicated nodes in the backbone network, while Oslo has four nodes in total. The second layer includes the NFV-MANOs and datacenter networks that have built-in NFVI that support NFV-VNFs. Moreover, the third layer is the NFV-VNFs.

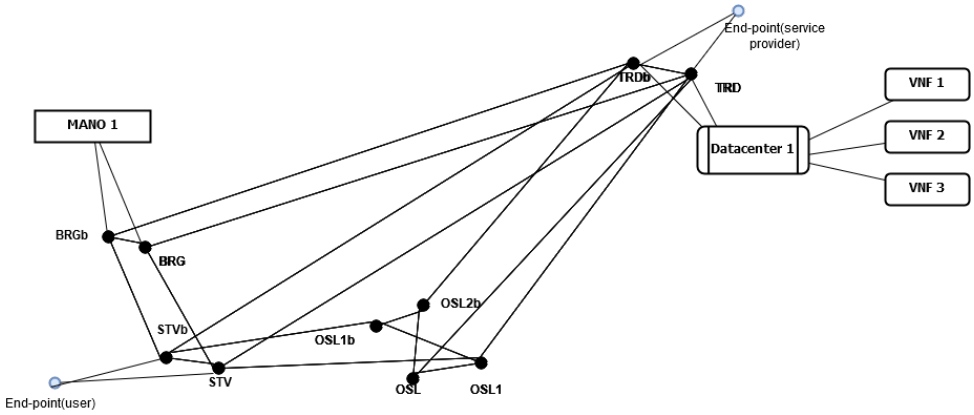

Figure 3.2: Reference scenario.

Significantly, since VNFs are just software instances based on NFVI, we present them as different nodes so that it will be presented in a more intuitional way. But the presented VNF nodes are just abstractions of the actual VNF instances. And the links in between datacenters and VNF nodes are considered as the actual VNF instances. MANO is considered as a whole component which includes both the underlying hardware infrastructure and the software running it.

Furthermore, besides the NFV elements, there are still two end-points in the network. One of the end-points is on the user side that is connected to STV and STVb nodes so that user can request and receive the service. The other end-point which connects with TRD and TRDb is on the service provider side which provides the service.

To obtain the minimal-cut set in an NFV-based network, there are four steps:

1 **Define the third-layer NFV-based network.**
Firstly, the Norwegian backbone network shall be defined as the first layer along with two end-points. Secondly, the datacenter network and the NFV elements shall be defined. Importantly, due to the different use cases, the geographical locations of datacenters and NFV elements may vary. Lastly, combine the defined networks and network elements together into the third-layer NFV-based network.

2 Find paths.
Based on the connectivity requirements on provisioning an NFV-service that have written in the beginning, the paths in between the different elements must be found. Including the paths start with the provider end-point and go to different VNFs in the service function chain and end up in the user end-point. Additionally, paths in between MANOs and the two endpoints as well as in between MANOs and VNFs must be found.

3 Map to the network elements.
These found paths need to be mapped into the vertexes of all the network elements such as links, VNFs, and etc. so that the failed component could be clearly located.

4 Calculate the minimum-cut sets.
Finally, the minimum cut sets of the NFV-based networks will be calculated.

After applying Mathematica software tool on Scenario 1, all the possible connections will be found and all the minimal-cut sets that exist will be obtained. Hence, the minimal-cut sets are shown in Table3-2 below and presented as different cardinality k. Each column represents the number of minimal-cut sets that have cardinality k. In addition, the sum contains the total number of minimal-cut sets.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	Sum
S_1	5	9	8	59	232	442	446	320	195	57	12	1785

Table 3.2: Minimal-cut sets of Scenario 1.

The highest cardinality is 11, it means there are at most 11 network elements that listed in the same minimal-cut set. If all of the 11 network elements are failed simultaneously will result in a system failure.

When the cardinality $k = 1$, there exists five network elements. It indicates that if just one network element among the five is not functioning anymore, the whole system considered as down, regardless how excellent other network elements in the system are working. These five crucial network elements are $l_{DC1-VNF1}$, $l_{DC1-VNF2}$, $l_{DC1-VNF3}$, N_{MANO1} and N_{DC1}. As we mentioned before, the link between a datacenter and a VNF is the actual VNF. Therefore, the three VNFs, MANO1 and Datacenter1 have a have a significant impact on the system vulnerability.

Similarly, the number of minimal-cut sets when cardinality $k = 2$ is 9. It means there are 9 sets of network elements and each of them includes 2 network elements. The 9 network elements are:

$$S1_{k=2} = \Big\{ \{n_{TRD}, n_{TRDb}\}, \{n_{STV}, n_{STVb}\}, \{n_{BRG}, n_{BRGb}\}, \{n_{TRD}, l_{TRDb-DC1}\}, \{n_{TRDb}, l_{TRD-DC1}\}, \{l_{TRDb-DC1}, l_{TRD-DC1}\}, \{n_{BRG}, l_{BRGb-MANO1}\}, \{n_{BRGb}, l_{BRG-MANO1}\}, \{l_{BRGb-MANO1}, l_{BRG-MANO1}\} \Big\}$$

From the minimal-cut sets, it can be seen that all the network elements are routers and physical links in the network. The analysis will be conducted in next chapter. However, with this in mind, the network elements that can influence the state of the systems generally can be categorized as five network elements. Namely, VNF, MANO, datacenter, router and link. Hence, let's take a further step into the reasons that make these network elements fail and the time it takes to recover from the different failures.

The structural analysis script is shown in Appendix A.

3.3 Dynamic model

After deploying the structure-function analysis of the network structure, the individual network element model shall be developed to evaluate the availability/unavailability. As mentioned before, the methods that can be applied are Markov model or Stochastic Petri net. For the purpose of this thesis work, the dynamic modeling approach is Stochastic Activity Network model which is one stochastic extension of Stochastic Petri nets [36]. The simulations and models were implemented in the Möbius software tool.

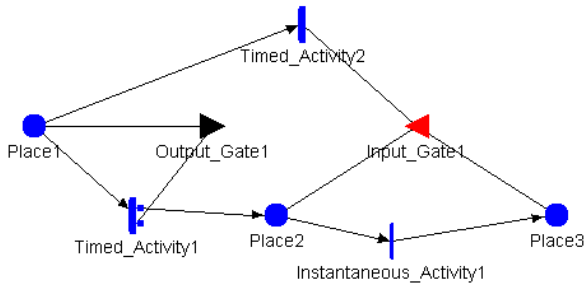

Figure 3.3: SAN system model example.

A simple example in Figure 3.3 is given above for demonstrating the SAN model. Stochastic activity network is a probabilistic extension of activity network which is comprised of 4 primitive objects: activities, places, input gates and output gates. Activities are about the actions happened in the modeled system. Places are the states of the modeled system. Input gates and output gates can be seen as controllers that control the activities. Furthermore, after an activity "completes", output gates are used to change the state of the system [7]. There are two types of activities: timed activities and instantaneous activities. A timed activity means the duration of the activity has an influence on the system's ability to perform. While an instantaneous activity of duration of 0 seconds related to the performance variable and is completed in a changeable period [36].

Moreover, the availability of the selected five network elements will be evaluated by use of SAN system models. These network elements were selected since they are all appearing in the lower cardinality minimal-cut sets. These five network element SAN models are link SAN model, router SAN model, datacenter SAN model, VNF SAN model and NFV-MANO SAN model. The SAN model of a link and a router are inspired by and taken from [34], [25], [9], [19]

A. Link

A link failure in our model is assumed to be related to the physical link failures. Hence, these links only have two states, either up or down due to hardware failures. Significantly, the aim is to provide a general model without considering the link length. The geographical location of the nodes could variate due to different cases so that the link length in between the different nodes will vary as well. Generally, the failure rate shall be proportional to the link length. However, in our model, we only consider that the link would fail if there is a hardware failure and the link length is not in our consideration [17].

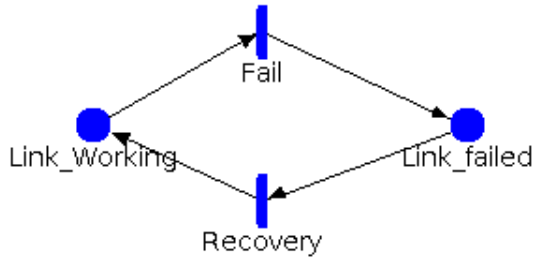

Figure 3.4: SAN model of a link.

The failure rate λ_L and repair rate μ_L are given in Table 3.3, and therefore the

availability of a link is "$A_L = \frac{\mu_L}{\lambda_L + \mu_L}$. All the failure rate, repair rate and the availability equation should be valid for all the links in the structure analysis model [34].

Parameter	Value	Description
λ_L	10^{-6}	Link failure rate
μ_L	0.01	Link recovery rate

Table 3.3: Link model parameters used in the case studies

B. Router

The purpose is to develop a SAN model for a general router despite the various types of router architecture. Therefore, the components in the router will not be considered in our case. In other words, we consider the router as a black box [34].

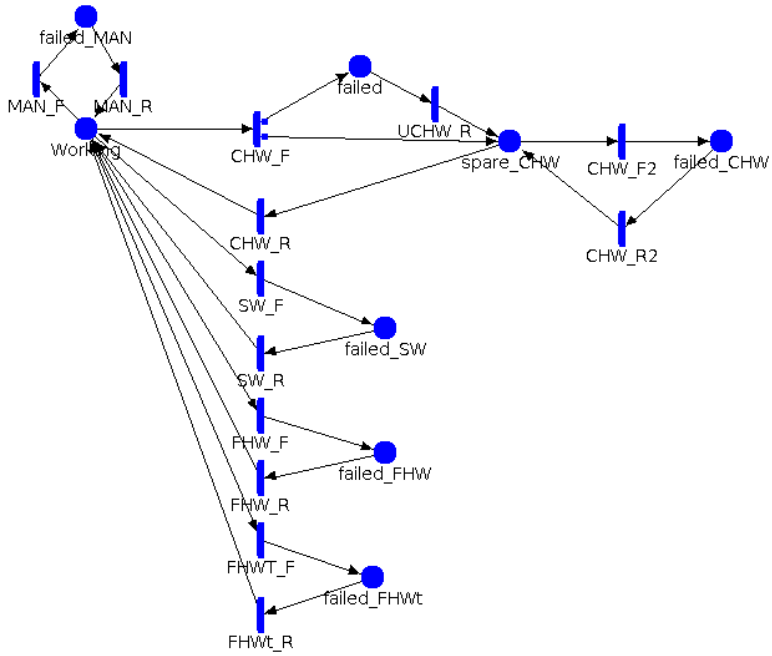

Figure 3.5: SAN model of a router.

The router model is based on a 1+1 redundancy router architecture. Additionally, there are a number of failures are not included in our model because they

considered as seldom happen and will not affect the expected accuracy of our model [34].

- Operation and Maintenance(O&M) failure. (failed_Man)
- Hardware failure on one controller. (failed/spare_CHW)
  When there is a hardware failure occurs on the working controller, the controller will fail. However, the redundancy controller either successfully activated or unsuccessfully triggered. If the redundancy controller is activated on time, the router will not fail. Vice versa.
- Hardware failures on both controllers. (failed_CHW)
- Permanent hardware failures in forwarding plane. (failed_FHW)
- Transient hardware failure in forwarding plane. (failed_FHWt)
- Software failure. (failed_SW)

Parameter	Value	Description
MAN_F	5.0E-7	Router O&M failure rate
MAN_R	9.0E-5	Router O&M recovery rate
CHW_F	0.97	Router hardware failure rate on one controller and the router will fail.
	0.03	Router hardware failure rate on one controller and the router will not fail.
UCHW_R	3.0E-5	Router hardware repair rate (on one controller)
CHW_F2	9.0E-9	Router hardware failure rate (on two controllers)
CHW_R2	2.0E-5	Router hardware recovery rate (on two controllers)
CHW_R	9.0E-9	Router hardware fail rate (on the redundancy controller)
SW_F	2.0E-6	Router software failure rate
SW_R	0.006	Router software recovery rate
FHW_F	9.0E-9	Router permanent hardware failure rate (in forwarding plane)
FHW_R	2.0E-5	Router permanent hardware repair rate (in forwarding plane)
FHWT_F	2.0E-6	Router transient hardware failure rate (in forwarding plane)
FHWt_R	0.006	Router transient hardware recovery rate (in forwarding plane)

Table 3.4: Router model parameters used in the case studies.

The SAN model and parameters can be applied in all the routers in our case. In addition, for the sake of simplicity, only homogeneous routers are considered.

C. Datacenter

Datacenter can be seen as a cluster of processors(servers) that have built-in NFVI to support NFV-VNFs [41]. NFVI are embedded in the datacenter and by using the provided physical resources as well as the software resources to provide the environment for VNF. To deploy the physical resources, these resources must be decoupled and abstracted by the virtualization layer to virtual resources. More specifically, the virtualization layer is based on a hypervisor to achieve this goal. These virtualized resources hence can be used in correspondent one or many Virtual Machine (VM) instances. Therefore, to consider the different types of failures that might cause datacenter unavailable, we must not only

consider the underlying infrastructure but also consider the virtualization layer and software upon it.
Here are some possible failures can occur in a datacenter infrastructure due to different reasons:

⇒ Network protocol
⇒ Hardware failures (link/server/rack failure)
⇒ Software failures
⇒ Battery failure (power failure)
⇒ Accidental/human error/misconfiguration
⇒ Water, heat failure
⇒ Weather related or nature disasters
⇒ Generator failure
⇒ Common O&M
⇒ Malicious attack

Generally, a datacenter should have redundant data communication connections to ensure the consistent communication. A backup power should be also supplied to ensure proper functioning of the datacenter. In addition, the environmental control should be equipped to prevent or reduce the risk that the datacenter will be brought down by the natural disaster or fire disaster and so on [16].
To provide a secure environment, a datacenter shall be highly integrated and the data shall be encrypted so that security issues such as malicious attack will be resisted. In a datacenter cluster, especially those with large-scale datacenter cluster, it is difficult to keep proper values for a large great deal of configuration parameters. Not to mention the "hidden" misconfigurations that could be revealed only when the system is not working properly. Thus, the misconfiguration detection method should be used in the datacenter to prevent this kind of failure [36]. Consequently, the possible risks in our case will be only between hardware failures and software failures.
As we mentioned in the previous chapter, there are three types of VNF deployment. All things considered, the second deployment approach (b) will be applied in our case for the sake of simplicity. In Figure **??** (b), VNF1 and VNF2 are sharing the same hardware. The underlying physical resources are divided to support two different VNFs. Since each VNF has a dedicated VM, the VM failure will not be considered as a failure resource in a datacenter(NFVI) in our case and will be considered in the VNF model development.
Therefore, in our case, the datacenter model only consists of the underlying hardware and the hypervisor. The supported VNFs will share the physical and virtual resources based on the same hardware and hypervisor. Notably, the specific components will not be considered in here and the datacenter is assumed as a black-box. Due to the different datacenter architectures, the SAN may vary, however, we only consider there is one server (hardware) in the datacenter with two types of failure resources showed in Figure 3.6. And the parameters are

shown in Table 3.5.

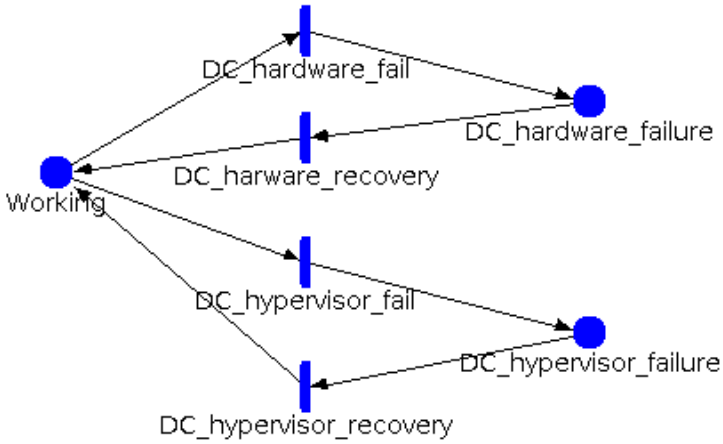

Figure 3.6: SAN model of a datacenter that support VNFs.

Parameter	Value	Description
DC_hardware_fail	0.97	Datacenter hardware failure rate
DC_hardware_recovery	9.0E-9	Datacenter hardware recovery rate
DC_hypervisor_fail	2.0E-6	Datacenter hypervisor failure rate
DC_hypervisor_recovery	0.006	Datacenter hypervisor recovery rate

Table 3.5: Datacenter model parameters used in the case studies

The SAN model and parameters can be applied in all the datacenters in our case. In addition, for the sake of simplicity, only homogeneous datacenters are considered. The datacenter model parameters showed above is inspired by and taken from [7]. Since the hypervisor is a software, we consider the failure and recovery rate as same as the general software failure rate and recovery rate.

D. VNF

As we mentioned before, VNFs are just software instances running in corresponding VMs. Therefore, in our case, we consider the VNFs and their correspondent VMs as a whole component. In the VNF model showed in Figure 3.7, there are only two failure source: VM failure and VNF software failure.

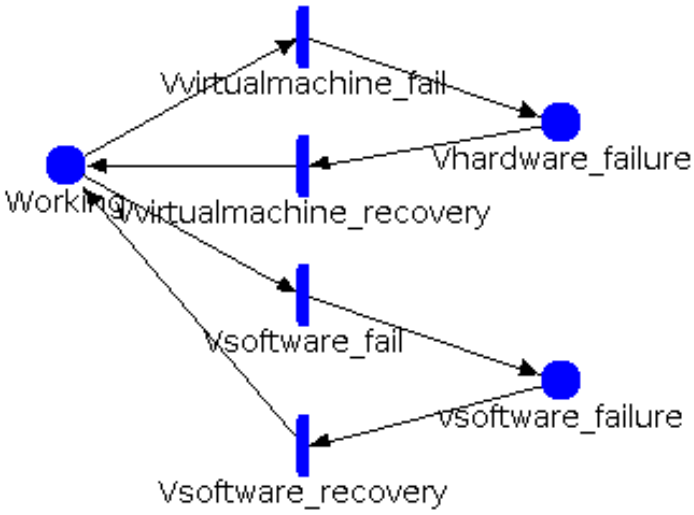

Figure 3.7: SAN model of a VNF.

Parameter	Value	Description
Vvirtualmachine_fail	2.0E-6	VNF virtual machine failure rate
Vvirtualmachine_recovery	0.006	VNF virtual machine recovery rate
Vsoftware_fail	2.0E-6	VNF software failure rate
Vsoftware_recovery	0.006	VNF software recovery rate

Table 3.6: VNF model parameters used in the case studies.

The SAN model and parameters can be applied in all the VNFs in our case. Since VM and VNF are software, we consider the failure and recovery rate as same as the general software failure rate and recovery rate. The VNF model parameters showed above is inspired by and taken from [7].

E. NFV-MANO

Since NFVI resources are supposed to support VNFs and partially VNFs, the MANO is not considered to be based on NFVI, but rather in a simple datacenter. For the sake of simplicity, the components inside the hardware will not be considered at all and there is no redundancy within the datacenter infrastructure. Hence, the SAN model of the MANO is shown in Figure 3.8.

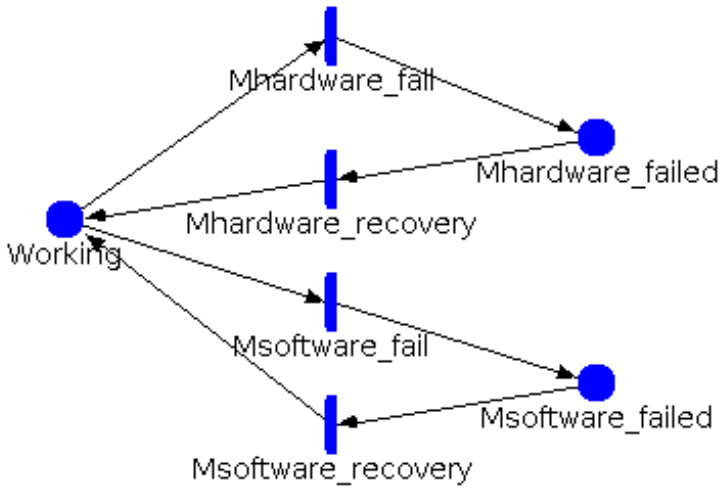

Figure 3.8: SAN model of a MANO.

Parameter	Value	Description
Mhardware_fail	0.97	MANO hardware failure rate
Mhardware_recovery	9.0E-9	MANO hardware recovery rate
Msoftware_fail	2.0E-6	MANO software failure rate
Msoftware_recovery	0.006	MANO software recovery rate

Table 3.7: MANO model parameters used in the case studies.

The SAN model and parameters can be applied in all the MANOs in our case. The MANO model parameters showed above is inspired by and taken from [7].

The unavailability of the five SAN models are presented below along with the 95% confidence interval.

	Unavailability
Link SAN	$1.0066 * 10^{-4} \pm 2.86 * 10^{-6}$
Router SAN	$6.0092 * 10^{-3} \pm 2.46 * 10^{-4}$
Datacenter SAN	$7.7881 * 10^{-4} \pm 7.56 * 10^{-5}$
VNF SAN	$6.5884 * 10^{-4} \pm 9.10 * 10^{-6}$
MANO SAN	$7.7881 * 10^{-4} \pm 7.56 * 10^{-5}$

Table 3.8: Comparison of the unavailability of the five network elements SAN model.

In addition, the documentations of the SAN models and on the simulation (reward and study) in Möbius will be presented in Appendix H-L.

3.4 Merge the two-level models

After achieving the minimal-cut sets from the structural models and the unavailability of network elements from the dynamic models, the overall network availability can be obtained by merging the results together by applying the inclusion-exclusion principle [37].

The inclusion-exclusion principle is a well-known instrumental in evaluating the system reliability of uncertainty over Boolean formulas [8]. The formula is:

$$A_S = P(\cup_{i=1}^{n} C_i) = \sum_{k=1}^{n} (-1)^{(k-1)} \sum_{\substack{\emptyset \neq \subseteq |n| \\ |l|=k}} P(\cap_{i \in I} C_i) \tag{3.1}$$

The C_i represents the minimal-cut sets that obtained from structural models. $P(Q_i)$ equals to the probability of set C_i [18].

The overall unavailability/availability are presented below.

Scenario	Unavailability	Availability
S_1	0.00232	0.99767

Table 3.9: The overall unavailability and availability of Scenario 1.

From the result, it can be seen that the NFV-based network service in Scenario 1 only achieved "two nines" which is far from the carrier grade services. Therefore, more scenarios will be conducted to evaluate the availability of NFV-based services in different network structures.

Moreover, the merging model script in Mathematica will be presented in Appendix M to Appendix S.

Chapter 4
Evaluation and analysis

As Chapter 3 illustrates the two-level availability approach used in obtaining the overall availability of the NFV-based services, one use case has been conducted as a reference scenario to show how the overall availability of the service can be achieved. Not surprisingly, the availability of the service still needs to improve to get to the carrier-grade standard.

In chapter 4, there are six more scenarios will be used to demonstrate the availability fluctuation when the NFV-elements are distributed in different NFVI-PoP. The structural analysis in different NFV-based network use cases will be illustrated along with the obtained minimal-cut sets. The evaluation and analysis will be conducted based on the use cases and related minimal-cut sets.

4.1 Six case studies

All the network scenarios below are based on the same underlying network structure which is the Norwegian backbone network. In the Reference scenario which presented in the last chapter, all the VNFs are running in the same datacenter that connects to TRD and TRDb nodes along with the MANO1 infrastructure. However, the VNFs and MANOs can be deployed either in the same geographical location or various geographical locations. NFV-MANO shall support geographically distributed deployment in order to complement the disaster recovery strategies [25]. Additionally, VNF instances can be also implemented either in the same location or be geographically dispersed when the overall end-to-end service performance and other policy regulations are met [21]. Therefore, six more network scenarios will be simulated to perform the distributed NFV deployment.

In Scenario 1, 2 and 3, there is only one MANO (MANO1) and the different VNFs are deployed either in the same datacenter or distributed in different datacenters. The same applies Scenario 4, 5 and 6, however, in these scenarios, two NFV-MANOs (MANO1 and MANO2) have been deployed.

A. Scenario 1

In Figure 4.1, there are 2 datacenters that locate in different geographical locations. Datacenter 1 connects with 2 nodes (TRD and TRDb), datacenter 3 connects to 4 nodes (OSL, OSL1, OSL1b and OSL2b). Both VNF1 and VNF2 are located in Datacenter 1 and VNF3 is located in Datacenter 3. MANO1 is connected to 2 nodes (BRG, BRGb) as depicted in Figure 4.1.

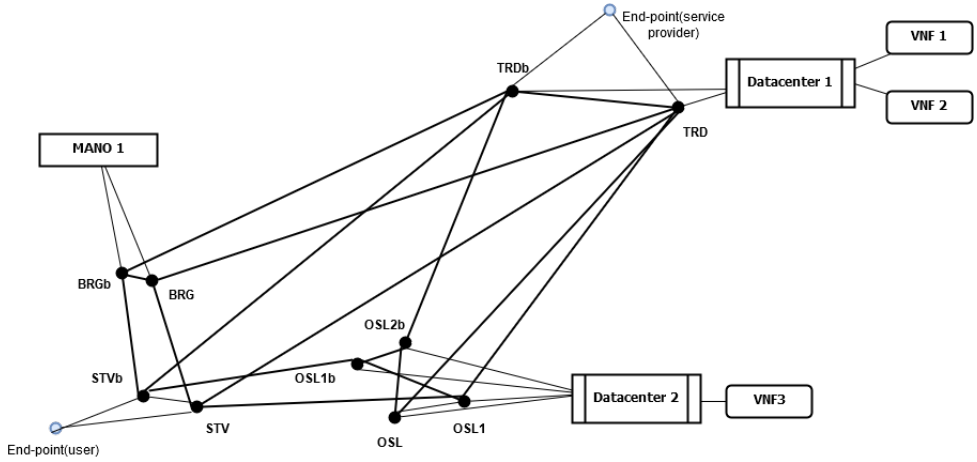

Figure 4.1: Depiction of Scenario 1.

After applying the structural analysis by going through all the possible connections in Scenario 1, the minimal-cut sets have been calculated as Table 4.1 showed.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	Sum
S_1	6	9	8	120	360	639	687	500	281	112	49	6	6	2783

Table 4.1: Minimal-cut sets of Scenario 1.

The highest cardinality is 13, as we explained in the last chapter, it means there are at most 13 network elements in one minimal-cut set. Together with the minimal-cut set with different cardinality value, the overall number of minimal-cut sets are also showed. S_1=2783. Compared to the Reference scenario, Scenario 1 has a more complex network structure since it has two datacenters which added 5 network elements in the system. Just because the added network elements, the total number of the minimal-cut sets are increasing, and the maximum cardinality is increasing as well.

B. Scenario 2

In Scenario 2 showed below, similarly, there are also two datacenters. The only difference is VNF2 are deployed by two datacenters (Datacenter 1, 2). And VNF1 and VNF3 locate in Datacenter 1 and Datacenter 3, respectively. MANO1 connects to 2 nodes (BRG and BRGb).

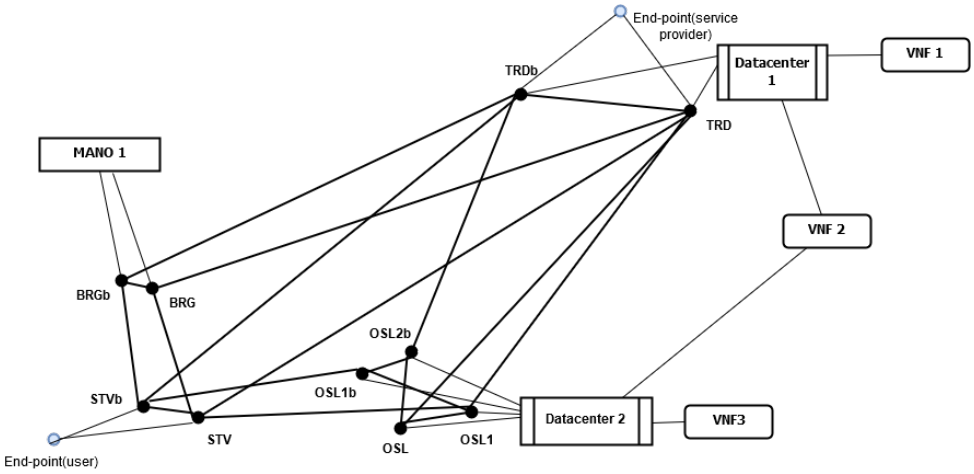

Figure 4.2: Depiction of Scenario 2.

After applying the structural analysis and going through all the possible connections in Scenario 2, the minimal-cut sets have been calculated as Table 4.2 showed.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	Sum
S_2	5	10	8	120	354	643	709	525	394	183	119	32	11	4	3117

Table 4.2: Minimal-cut sets of Scenario 2.

The highest cardinality is 14 and the overall number of minimal-cut sets are S_2=3117. Scenario 1 and Scenario 2 only have one difference, a link in between VNF2 and Datacenter 2. As we explained earlier in the last chapter, the node VNF2 is considered as fault-free since it is just representing the abstraction of VNF2. The real VNF2 software instance replications are presented by using links in between node VNF2 and datacenters. Even there is only one extra link in Scenario 2, the overall possible connections are increasing which result in the growth of the total number of the minimal-cut sets and the maximum cardinality in a minimal-cut set.

C. *Scenario 3*

In Scenario 3, a new datacenter (Datacenter 3) is added into the NFV-based network structure. The newly added datacenter connected with BRG, BRGb, STV and STVb. VNF1 still locates in Datacenter 1. VNF2 are owned by two datacenters (Datacenter 1, 2) just as Scenario 2 showed. And VNF3 has replications in both Datacenter2 and Datacenter3. MANO1 is still in the same location as the previous scenarios presented before.

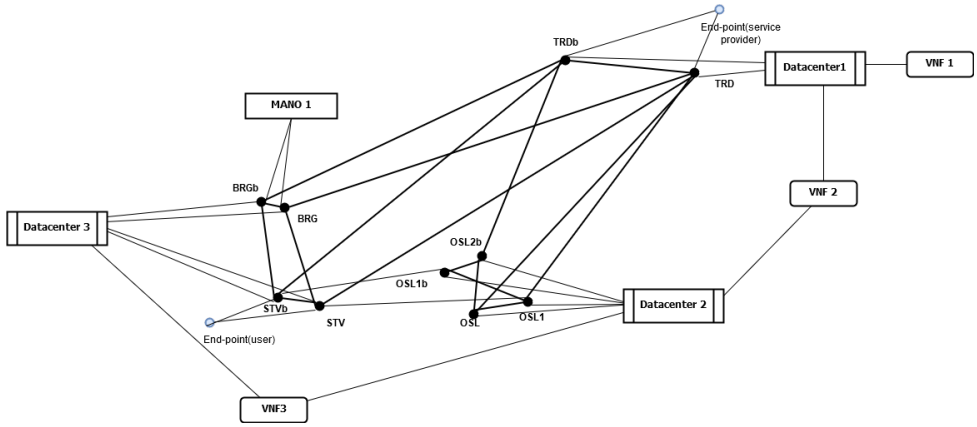

Figure 4.3: Depiction of Scenario 3.

After applying structural analysis in Scenario 3, the minimal-cut sets have been calculated as Table 4.3 showed.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	Sum
S_3	3	15	8	65	431	766	1038	1389	1431	1619	1559	1198	875	532	268	142	60	13	3	11415

Table 4.3: Minimal-cut sets of Scenario 3.

The overall number of minimal-cut sets are S_3=11415 which is exceedingly larger than the previous 3 scenarios and the highest cardinality is 19. Scenario 3 is the most complex scenario so far, there are 6 extra network elements have been added to the network compared with Scenario 3 and 12 extra network elements compared to the Reference scenario.

D. *Scenario 4*

Unlike the three scenarios showed above, the last three scenarios all have two redundant MANOs in different geographical locations to manage and orchestrate VNFs. Since there are two MANOs, so at least one MANO must be reachable from all the VNFs and all the endpoints. MANO 1 connects to BRG node and

BRGb node. MANO2 connects to TRD node and TRDb node just as Datacenter 1 does. Datacenter2 is connected with four nodes: OSL, OSL1, OSL1b and OSL2b. VNF1 and VNF2 are running in Datacenter1, while VNF3 locates in Datacenter2.

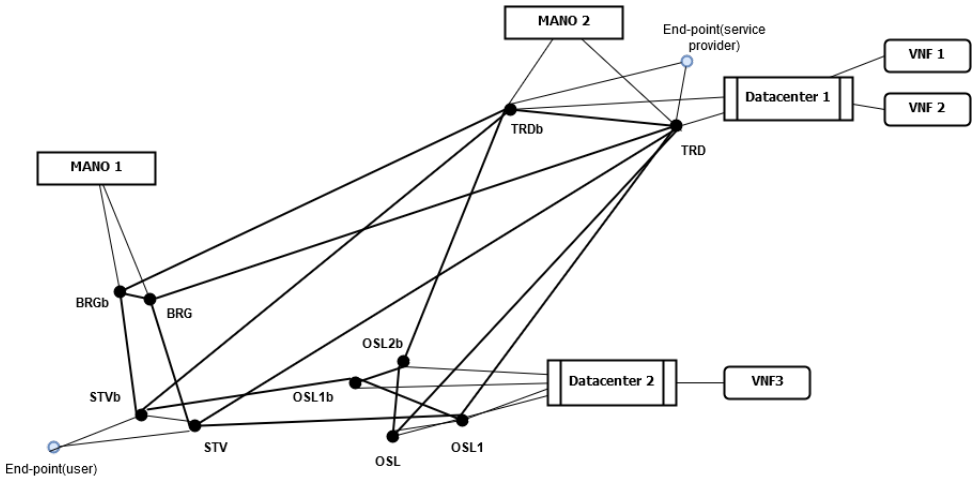

Figure 4.4: Depiction of Scenario 4.

After going through all the possible connections in Scenario 4, the minimal-cut sets have been calculated as Table 4.5 showed and the overall number of minimal-cut sets are S_4=6474.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	Sum
S_4	5	6	7	121	335	668	1251	1339	1157	752	451	236	80	51	10	5	6474

Table 4.4: Minimal-cut sets of Scenario 4.

The only difference of the network structure in Scenario 4 and Scenario 1 is that one NFV-MANO (MANO2) is added. Consequently, three network elements have been added to the network structure in Scenario 4. Thus, due to the added complexity the overall number of minimal-cut sets has increased to 6474 from 2783.

E. Scenario 5

In Scenario 5, VNF2 has replications in 2 datacenters: Datacenter 1 and Datacenter 2. Other than that, everything is the same as Scenario 4.

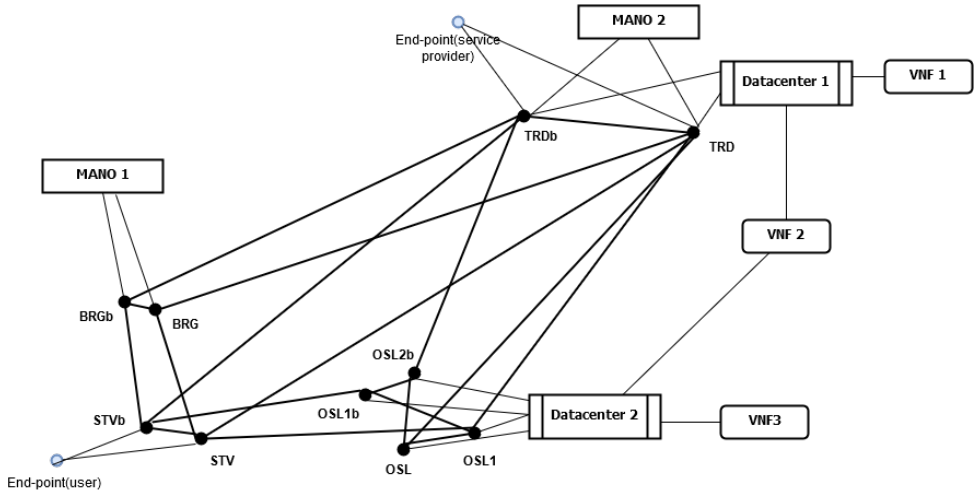

Figure 4.5: Depiction of Scenario 5.

After going through all the possible connections in Scenario 5, the minimal-cut sets have been calculated as Table 4.5 showed. The overall number of minimal-cut sets are S_5=8284.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	Sum
S_5	4	7	7	121	323	628	1159	1492	1537	1225	898	476	262	83	48	9	5	8284

Table 4.5: Minimal-cut sets of Scenario 5.

Scenario 5 has only added one extra network element based on the network structure in Scenario 4, the link between Datacenter2 and VNF2. It leads to an increase in all the possible connections in Scenario 5 and hence results in the growth on the sum of the number of minimal-cut sets.

F. Scenario 6

Scenario 6 has the most complex network structure. There are two redundant MANOs and three datacenters. Two of the different VNFs have replications in multiple datacenters.

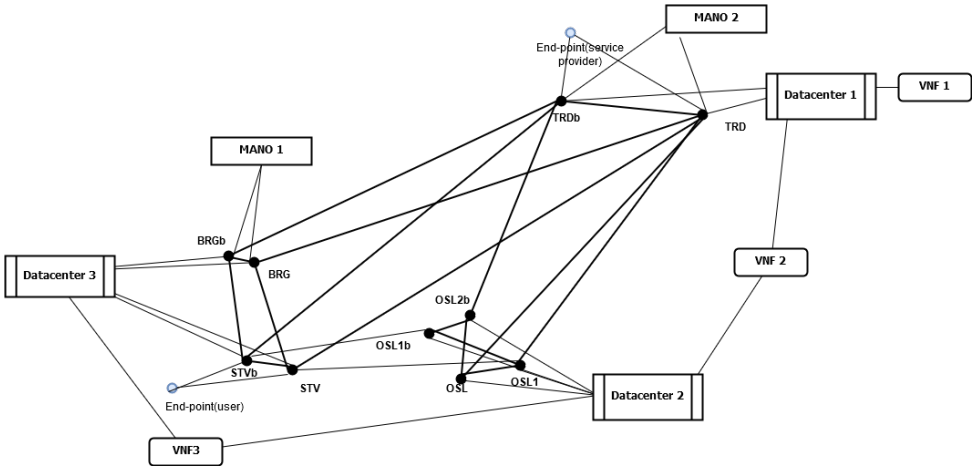

Figure 4.6: Scenario 6 of the structural analysis.

After going through all the possible connections in Scenario 6, the minimal-cut sets have been calculated as Table 4.6 showed.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	Sum
S_6	2	12	7	66	432	793	1477	2605	3708	5136	5714	3576	2367	1388	730	344	117	35	6	33372

Table 4.6: Minimal-cut sets of Scenario 6.

The overall number of minimal-cut sets are S_6=33372. Noteworthily, the sum number of all the minimal-cut sets is the largest among the seven scenarios, and the network structure is the most complex one. The highest cardinality is 19, meaning, if the whole 19 network elements in a minimal-set are failed simultaneously regardless the reason, the system will fail.

4.2 Evaluation and analysis of the seven case studies

4.2.1 Comparison of the minimal-cut sets in the seven scenarios

There are 7 scenarios have been conducted to present the minimal-cut sets of all the possible connections in delivering NFV-based services. The number of the overall minimal-cut sets varied depending on the network structure of each scenario. As we explained in Chapter 3, the state of the network element in a minimal-cut set can decide the state of the whole system, meaning whether the system is functioning and whether the service will be delivered to the dedicated user are depending on the network elements in the minimal-cut sets. If all the subsystems in the minimal-cut

set are failed, the system considered as fail and the service can not be delivered to the users.

As the seven use cases demonstrated, when the network structure gets more complex, the cardinality will increase accordingly and so is the sum of the minimal-cut sets. And as we stated earlier, in Scenario 6, the maximum number of sub-elements among all the minimal-cut sets is 19. That is to say, the 19 sub-elements in a minimal-cut set are failed at the same time can result in a system failure. This is a rare case in reality. Therefore, we only consider the low cardinality sets (C1-C4) in our thesis work since they are most likely to happen. The number of low cardinality cut sets get higher, then the system is easier to fail. We regard the fewer simultaneous network elements failures to indicate the vulnerability of the network. The low cardinality minimal-cut sets of the seven scenarios are presented below along with the sum of the first four low cardinalities sets and the overall minimal-cut sets in each scenario.

		C_1	C_2	C_3	C_4	Sum ($C_1 + C_2 + C_3 + C_4$)	Sum (overall)
R-S		5	9	8	59	81	1785
One MANO	S_1	6	9	8	120	143	2783
	S_2	5	10	8	120	143	3117
	S_3	3	15	8	65	91	11415
Two MANOs	S_4	5	6	7	121	139	6474
	S_5	4	7	7	121	139	8284
	S_6	2	12	7	66	87	33372

Table 4.7: Comparison of the different scenarios' low cardinality minimal-cut sets.

The Reference scenario has the simplest network structure only with 10 nodes, 1 datacenter, 3 VNFs and 1 MANO. Just because of the simple network structure, the first four cardinality number of the minimal-cut sets is only 81 which is the smallest number of the 7 scenarios. It indicates the network structure in the Reference scenario is the least vulnerable one. The rest of the scenarios are just variations based on the Reference Scenario.

Scenario 3 and 6 also have smaller numbers in the sum of first four cardinality sets, however, both network structures are the most complex network structures among the 7 use cases. Therefore, it reveals that adding complexity into the network structure may not always have a bad impact on the service dependability. To achieve relatively dependable network service by deploying NFV elements, the elements in the low cardinality minimal-cut sets shall be listed and analyzed.

	One MANO				Two MANOs		
C_1 element	R-S	S_1	S_2	S_3	S_4	S_5	S_6
	$\{l_{DC1-VNF1}\}$	$\{l_{DC1-VNF1}\}$	$\{l_{DC1-VNF1}\}$	$\{l_{DC1-VNF1}\}$	$\{l_{DC1-VNF1}\}$	$\{l_{DC1-VNF1}\}$	$\{l_{DC1-VNF1}\}$
	$\{l_{DC1-VNF2}\}$	$\{l_{DC1-VNF2}\}$	$\{l_{DC2-VNF3}\}$	$\{N_{MANO1}\}$	$\{l_{DC2-VNF2}\}$	$\{l_{DC2-VNF3}\}$	$\{N_{DC1}\}$
	$\{l_{DC1-VNF3}\}$	$\{l_{DC2-VNF3}\}$	$\{N_{MANO1}\}$	$\{N_{DC1}\}$	$\{l_{DC2-VNF3}\}$	$\{N_{DC1}\}$	
	$\{N_{MANO1}\}$	$\{N_{MANO1}\}$	$\{N_{DC1}\}$		$\{N_{DC1}\}$	$\{N_{DC2}\}$	
	$\{N_{DC1}\}$	$\{N_{DC1}\}$	$\{N_{DC2}\}$		$\{N_{DC2}\}$		
		$\{N_{DC2}\}$					
Sum	5	6	5	3	5	4	2

Table 4.8: Comparison of elements of minimal-cut sets when cardinality=1 in all the scenarios.

In Table 4.8, all the network elements of in minimal-cut sets when cardinality is 1 in each scenario are listed. When cardinality=1, one of the elements' failure will cause the whole system state change. More formally, the service will not be provided to the user. From Table 4.8, it can be clearly seen that the number of sets starts to decrease when the NFV elements have replications. Specifically, the number in Scenario 2 and Scenario 3 is smaller than Scenario 1. Similarly, the number in Scenario 5 and Scenario 6 is smaller than all the other scenarios.

In the Reference scenario, all the VNFs are in the same datacenter which is very risky. If the datacenter is failed regardless the reason, the service will not be provided to its subscriber. In scenario 1, there are two datacenters, but similarly, all the NFV elements have no backup replications in any other datacenter, so that one link failure or one datacenter failure will cause the whole system out of service. From the observation, it can be seen if NFV-elements do not have any replications or redundancy in the datacenter, adding a new extra datacenter only increase the network vulnerability.

In scenario 2, VNF2 are deployed by two datacenters, so the number of minimal-cut sets with cardinality one decreases to 5 which is the same number as in Reference scenario. In addition, in scenario 3, VNF2 and VNF3 are both deployed by two different datacenters. The same results apply to the scenarios with 2 MANOs, there are only two network elements exists in Scenario 6. It indicates that having a backup or replications of NFV elements in the network make the network less vulnerable.

Since the datacenters, MANOs are connected with at least 2 nodes to access the backbone network. Another investigation shall be made based on the minimal-cut sets with cardinality 2. When cardinality=2, it means if the two listed network elements failed simultaneously, the system will fail. If only one element in the set failed, the system is still in its working state. The reason for this type of result is

because of the redundancy, i.e. two nodes connected with an endpoint or a MANO or a datacenter. And in Scenario 4, 5, 6, there are also redundancy of MANOs.

	One MANO			
C_2 element	R-S	S_1	S_2	S_3
	$\{n_{TRD}, n_{TRDb}\}$	$\{n_{TRD}, n_{TRDb}\}$	$\{n_{TRD}, n_{TRDb}\}$	$\{n_{TRD}, n_{TRDb}\}$
	$\{n_{STV}, n_{STVb}\}$	$\{n_{STV}, n_{STVb}\}$	$\{n_{STV}, n_{STVb}\}$	$\{n_{STV}, n_{STVb}\}$
	$\{n_{BRG}, n_{BRGb}\}$	$\{n_{BRG}, n_{BRGb}\}$	$\{n_{BRG}, n_{BRGb}\}$	$\{n_{BRG}, n_{BRGb}\}$
	$\{n_{TRD}, l_{TRDb-DC1}\}$	$\{n_{TRD}, l_{TRDb-DC1}\}$	$\{n_{TRD}, l_{TRDb-DC1}\}$	$\{n_{TRD}, l_{TRDb-DC1}\}$
	$\{n_{TRDb}, l_{TRD-DC1}\}$	$\{n_{TRDb}, l_{TRD-DC1}\}$	$\{n_{TRDb}, l_{TRD-DC1}\}$	$\{n_{TRDb}, l_{TRD-DC1}\}$
	$\{l_{TRDb-DC1}, l_{TRD-DC1}\}$	$\{l_{TRDb-DC1}, l_{TRD-DC1}\}$	$\{l_{TRDb-DC1}, l_{TRD-DC1}\}$	$\{l_{TRDb-DC1}, l_{TRD-DC1}\}$
	$\{n_{BRG}, l_{BRGb-MANO1}\}$	$\{n_{BRG}, l_{BRGb-MANO1}\}$	$\{n_{BRG}, l_{BRGb-MANO1}\}$	$\{n_{BRG}, l_{BRGb-MANO1}\}$
	$\{n_{BRGb}, l_{BRG-MANO1}\}$	$\{n_{BRGb}, l_{BRG-MANO1}\}$	$\{n_{BRGb}, l_{BRG-MANO1}\}$	$\{n_{BRGb}, l_{BRG-MANO1}\}$
	$\{l_{BRGb-MANO1}, l_{BRG-MANO1}\}$	$\{l_{BRGb-MANO1}, l_{BRG-MANO1}\}$	$\{l_{BRGb-MANO1}, l_{BRG-MANO1}\}$	$\{l_{BRGb-MANO1}, l_{BRG-MANO1}\}$
			$\{l_{VNF2-DC1}, l_{VNF2-DC2}\}$	$\{l_{VNF2-DC1}, l_{VNF2-DC2}\}$
				$\{l_{VNF3-DC3}, l_{VNF3-DC2}\}$
				$\{n_{DC2}, n_{DC3}\}$
				$\{n_{DC2}, l_{DC1-VNF2}\}$
				$\{n_{DC2}, l_{DC3-VNF3}\}$
Sum	9	9	10	15

	Two MANOs		
C_2 element	S_4	S_5	S_6
	$\{n_{TRD}, n_{TRDb}\}$	$\{n_{TRD}, n_{TRDb}\}$	$\{n_{TRD}, n_{TRDb}\}$
	$\{n_{STV}, n_{STVb}\}$	$\{n_{STV}, n_{STVb}\}$	$\{n_{STV}, n_{STVb}\}$
	$\{n_{TRD}, l_{TRDb-DC1}\}$	$\{n_{TRD}, l_{TRDb-DC1}\}$	$\{n_{TRD}, l_{TRDb-DC1}\}$
	$\{n_{TRDb}, l_{TRD-DC1}\}$	$\{n_{TRDb}, l_{TRD-DC1}\}$	$\{n_{TRDb}, l_{TRD-DC1}\}$
	$\{n_{MANO1}, n_{MANO2}\}$	$\{n_{MANO1}, n_{MANO2}\}$	$\{l_{TRDb-DC1}, l_{TRD-DC1}\}$
	$\{l_{TRDb-DC1}, l_{TRD-DC1}\}$	$\{l_{TRDb-DC1}, l_{TRD-DC1}\}$	$\{l_{VNF2-DC1}, l_{VNF2-DC2}\}$
		$\{l_{VNF2-DC1}, l_{VNF2-DC2}\}$	$\{l_{VNF3-DC3}, l_{VNF3-DC2}\}$
			$\{n_{DC2}, n_{DC3}\}$
			$\{n_{DC2}, l_{DC1-VNF2}\}$
			$\{n_{DC2}, l_{DC3-VNF3}\}$
			$\{n_{DC3}, l_{DC2-VNF3}\}$
			$\{n_{MANO1}, n_{MANO2}\}$
Sum	6	7	12

Table 4.9: Comparison of elements of minimal-cut sets when cardinality=2 in all the scenarios.

Table 4.9 listed the network elements in minimal-cut sets with cardinality 2.

Even though Scenario 4, 5, 6 has a smaller number than Scenario 1, 2, 3. It does not reveal the fact that having two redundant MANO can achieve a more dependable network structure since the added MANO2 are connected the same nodes as the service provider endpoint *o* and Datacenter 1 connected. However, in other word, it shows that adding new NFV-elements that locate as same as the existing network

elements may lead the network structure less vulnerable.

The network elements in minimal-cut sets with cardinality 4 and 5 will not be presented for the sake of simplicity. In the Reference scenario, we conducted a situation that all the VNFs are running in the same datacenter. This may lead to a less vulnerable situation than any other scenarios we have conducted. However, this deployment is highly counting on the dependability of the datacenter. If the datacenter crashes for any reason, the services can not be provisioned anymore. And putting everything in one datacenter is not practical due to the geographical restrictions.

In [14], there are suggestions from ETSI that the NFV deployment should have a geographical distribution for the failure recovery issues. Therefore, Scenario 1 and 4 have been conducted that the VNFs are distributed in different geographic locations, but none of them are hosted by more than one datacenter. While in the rest of the scenarios, there are at least one VNF is deployed by more than two datacenters in different geographic locations. The results are obvious that Scenario 1 and 4 provide the most vulnerable network structure. Thus, there are some suggestions just based on the structural analysis of the seven scenarios.

- If the datacenter is used only to provide one single VNF without replications, then it is not necessary to use this datacenter, but rather use the existing datacenters to provide. Just as Scenario 2 showed, Datacenter 2 is used only to support VNF3 and the network becomes more vulnerable than the original (Reference scenario).
- If the datacenter is used to deploy multiple VNFs and especially for those VNFs have replications in other datacenters, the network structure tends to be less vulnerable than the original, just as Scenario 3 and Scenario 6 showed.
- If a new NFV-element will add into the network structure, it is better to connect the element with the nodes that already connected with other NFV-element or the endpoints, just as MANO2 in Scenario 4, 5 and 6 showed.

In spite of the low cardinality minimal-cut sets are used to reflect the network vulnerability, the structural analysis of the network scenario is not sufficient to achieve the network availability evaluation. The overall service unavailability/availability must be included since the failure rate and repair rate of different network elements are various.

4.2.2 Comparison of the availability of the NFV-based services

Merging the obtained minimal-cut sets and the unavailability that achieved from the dynamic SAN models in the previous chapter, the overall unavailability of the service in different NFV-based networks are obtained and presented in Table 4.10.

	Scenario	Unavailability	Availability
One MANO	S_1	0.002326	0.997674
	S_2	0.004415	0.995585
	S_3	0.00376	0.99624
	S_4	0.002329	0.99671
Two MANOs	S_5	0.003046	0.996954
	S_6	0.00239	0.99761
	S_7	0.001514	0.998486

Table 4.10: Comparison of the unavailability/availability of the NFV-based service in different scenarios.

The availability of the services provided in 7 different NFV-based networks have the same pattern as the number of the minimal-cut sets in lower cardinalities. It is mainly because of the SAN model results. For the sake of simplicity and limited time, only the general and dominant failure resources are considered in the modelling process. Therefore, the unavailability of the network elements did not influence the result too much.

Firstly, the availability of the service in all the scenarios has not reached even "3 nines". In Chapter 2, we have mentioned that the carrier-grade service always achieves the "5 nines" standard or even beyond the "5 nines". The idea of deploying NFV technology is to adding complexity to the network to handle complexity. By applying NFV technology in the existing network, the dependency on the specialized hardware will greatly reduce and increase the service agility. But if the service dependability does not reach the standard, then the technology is not ready to be used.

Moreover, in all the scenarios, the NFV-based service availability model only considers the network connectivity aspects. There are still many correlation failure resources exist in the real deployment that may further influence the availability of the NFV-based services.

Secondly, having VNF replications in two datacenters can increase the service dependability. Scenario 2, 3, 5 and 6 are proof of this. Distributed NFV deployment can take advantage of the cloud computing as well. The VNFs might host and managed by different infrastructure providers in the cloud, and the service providers rent the needed VNFs for service provisioning.

Thirdly, adding two MANO redundancies can increase the service dependability. But how much they increased are debatable, since in our case MANO2 is taking

advantage of the existing node connections. More than two MANO redundancies have not simulated in our case. However, adding MANO redundancies to increase the service dependability may have a threshold. If there are too many MANO redundancies, the service dependability might decrease due to the increasing network elements in the network structure.

As a conclusion, the approach of deploying the NFV-element in the telecommunication network to provide services still need to be improved so that its service availability can reach carrier grade. It is also a trade-off among economic input, the network complexity and the dependability. To achieve a better service availability, more links and datacenters will be used, and more expensive it is. However, if the economy does not allow, put all the VNFs in one datacenter and deploy the NFV-MANO in the same nodes that connect with either the datacenter or the endpoints to achieve a higher availability. In contrast, if money allows, deploy redundancy NFV-MANO and VNFs to achieve the higher availability. But how many replications and redundancies should be deployed is a debatable question and is highly depending on the realistic network situation.

Chapter 5
Conclusion and future work

5.1 Conclusion

As NFV plays a significant role in changing the way of provisioning network services, it is important for TSPs to know the challenges, risks as well as the influences on the existing telecommunication infrastructures. However, there are still many unexplored areas in NFV need to be studied. For example, the dependability concern in NFV. Therefore, this thesis work is focusing on the service dependability in NFV-based networks.

The two-level availability model approach has been applied in the seven different NFV-based Norwegian backbone network scenarios. By utilizing the two-level availability model, both the network structure and the network elements behavior in the structure can be fully studied and simulated to present the impacts of the service dependability from various NFV deployment. The main contributions of the thesis work are:

- Provide a literature study with regard to the NFV architecture and NFV deployment.
- Illustrate the way to apply the two-level availability model approach along with seven scenarios to obtain the minimal-cut sets and the overall dependability of NFV-based services.
- Analysis the obtained results and evaluate the impact of the NFV deployment on the service availability.

This thesis work is a step towards the dependability concerns regarding NFV. In order to gain insight, the two-level availability model approach is taken, where the goal is to evaluate the impact of the NFV deployment on the service dependability. Using NFV technology in the current service provisioning network will add complexity and bring unknown challenges, risks in the network. Therefore, it is important to be aware of the potential side effects brings by the NFV deployment. In the scenarios as we developed, adding datacenter to host single VNF software instance may reduce

the service dependability while adding VNF replications and NFV-MANO redundant may increase the service dependability. It is always a trade-off between complexity and dependability and adding complexity into the network to handle complexity might increase the risks and at the same time reduce the service dependability.

5.1.1 Future work

Due to the limited time and for the sake of simplicity, there are several things could be done in the future. Firstly, the approach and the evaluation are only applied in the Norwegian backbone network which is a simple but realistic scenario. However, it also can be applied in a world-wide backbone network to perform the service availability changes when the network topology becomes more complex. Secondly, the network services, in our scenarios, are only composed of three VNFs. However, in reality, the network service function chain might have more network functions involved in it. Lastly, the SAN models of datacenter, VNF and MANO are very simple and only have limited failure resources.

References

[1] Carrier grade. https://en.wikipedia.org/wiki/Carrier_grade.

[2] Cloudnfv. http://cloudnfv.com/.

[3] A comparison of nfv, sdn and cloud computing. http://telecomdrive.com/comparison-nfv-sdn-cloud-computing/.

[4] Huawei nfv open lab. http://pr.huawei.com/en/news/hw-411889-nfv.htm#.WULZVsaF9sQ.

[5] Möbius model-based environment for validation of system reliability, availability, security, and performance. https://www.mobius.illinois.edu/.

[6] Network functions virtualisation. http://www.etsi.org/technologies-clusters/technologies/nfv.

[7] San atomic formalism. https://www.mobius.illinois.edu/wiki/index.php/SAN_Atomic_Formalism.

[8] F. Aguirre, S. Destercke, D. Dubois, M. Sallak, and C. Jacob. Inclusion–exclusion principle for belief functions. *International Journal of Approximate Reasoning*, 55(8):1708 – 1727, 2014.

[9] Richard E Barlow and Frank Proschan. Statistical theory of reliability and life testing: probability models. Technical report, DTIC Document, 1975.

[10] D. Cotroneo, L. De Simone, A. K. Iannillo, A. Lanzaro, and R. Natella. Dependability evaluation and benchmarking of network function virtualization infrastructures. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–9, April 2015.

[11] Elena Dubrova. *Fault-Tolerant Design*. Springer-Verlag New York, 2013.

[12] PJ Emstad, Poul E Heegaard, Bjarne E Helvik, and L Paquereau. Dependability and performance in information and communication systems-fundamentals, 2008.

[13] A. Gonzalez, P. Gronsund, K. Mahmood, B. Helvik, P. Heegaard, and G. Nencioni. Service availability in the nfv virtualized evolved packet core. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2015.

[14] Andres J. Gonzalez and Bjarne E. Helvik. Characterisation of router and link failure processes in uninett's ip backbone network, 2012.

[15] P. Mell T. Grance. The nist definition of cloud computing. Technical report, National Institute of Standards and Technology.

[16] Peter J. Haas. *Stochastic Petri Nets*. Springer New York, 2002.

[17] P. E. Heegaard, B. E. Helvik, and V. B. Mendiratta. Achieving dependability in software-defined networking; a perspective. In *2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM)*, pages 63–70, Oct 2015.

[18] Poul E. Heegaard, Bjarne E. Helvik, Gianfranco Nencioni, and Jonas Wäfler. *Managed Dependability in Interacting Systems*, pages 197–226. Springer International Publishing, Cham, 2016.

[19] H. Heffes and D. Lucantoni. A markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. *IEEE Journal on Selected Areas in Communications*, 4(6):856–868, Sep 1986.

[20] F. Hu, Q. Hao, and K. Bao. A survey on software-defined network and openflow: From concept to implementation. *IEEE Communications Surveys Tutorials*, 16(4):2181–2206, Fourthquarter 2014.

[21] ETSI NFV ISG. Network functions virtualisation (nfv) architectural framework. Technical report, ETSI, October 2013.

[22] ETSI NFV ISG. Network functions virtualisation (nfv) use cases. Technical report, ETSI, October 2013.

[23] ETSI NFV ISG. Network functions virtualisation (nfv) management and orchestration. Technical report, ETSI, December 2014.

[24] ETSI NFV ISG. Network functions virtualisation (nfv) terminology for main concepts in nfv. Technical report, ETSI, December 2014.

[25] ETSI NFV ISG. Network functions virtualisation (nfv) resiliency requirements. Technical report, ETSI, January 2015.

[26] ETSI NFV ISG. Network functions virtualisation (nfv); management and orchestration; report on architectural options. Technical report, ETSI, July 2016.

[27] R. Kumar. *Research Methodology: A Step-by-Step Guide for Beginners*. SAGE Publications, 2010.

[28] Jean-Claude Laprie. Dependable computing and fault-tolerance. *Digest of Papers FTCS-15*, pages 2–11, 1985.

[29] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba. Topology-aware prediction of virtual network function resource requirements. *IEEE Transactions on Network and Service Management*, 14(1):106–120, March 2017.

[30] R. Mijumbi, J. Serrat, and J. L. Gorricho. Self-managed resources in network virtualisation environments. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1099–1106, May 2015.

[31] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262, Firstquarter 2016.

[32] R. Mijumbi, J. Serrat, J. l. Gorricho, S. Latre, M. Charalambides, and D. Lopez. Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54(1):98–105, January 2016.

[33] R. Mijumbi, J. Serrat, J. l. Gorricho, S. Latre, M. Charalambides, and D. Lopez. Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54(1):98–105, January 2016.

[34] G. Nencioni, B. E. Helvik, A. J. Gonzalez, P. E. Heegaard, and A. Kamisinski. Availability modelling of software-defined backbone networks. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 105–112, June 2016.

[35] M. R. Sama, L. M. Contreras, J. Kaippallimalil, I. Akiyoshi, H. Qian, and H. Ni. Software-defined control of the virtualized mobile packet core. *IEEE Communications Magazine*, 53(2):107–115, Feb 2015.

[36] William H. Sanders and John F. Meyer. *Stochastic Activity Networks: Formal Definitions and Concepts*, pages 315–343. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[37] Todd Silvestri. Complex system reliability; a graph theory approach. *The Mathematica Journal*, 16, July 2014.

[38] Ricard Vilalta, Raül Mu noz, Arturo Mayoral, Ramon Casellas, Ricardo Martínez, Víctor López, and Diego López. Transport network function virtualization. *J. Lightwave Technol.*, 33(8):1557–1564, Apr 2015.

[39] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. Characterizing cloud computing hardware reliability. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 193–204. ACM, 2010.

[40] Wikibooks. Communication networks/network basics — wikibooks, the free textbook project. https://en.wikibooks.org/w/index.php?title=Communication_Networks/Network_Basics&oldid=3230653, 2017.

[41] Wolframalpha. Mathematica. https://www.wolfram.com/mathematica/.

[42] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

Appendix A

Structural analysis script of Reference scenario in Mathematica

Structure model of Reference scenario

Define network

```
Primary = Graph[{TRD ⟷ OSL1, TRD ⟷ OSL2,
    TRD ⟷ BRG, TRD ⟷ STV, OSL1 ⟷ OSL2, OSL1 ⟷ STV, STV ⟷ BRG}];
Secondary = Graph[{TRDb ⟷ OSL2b, TRDb ⟷ BRGb, TRDb ⟷ STVb,
    OSL1b ⟷ OSL2b, OSL1b ⟷ STVb, STVb ⟷ BRGb}];
ConnectionPS = Graph[{TRD ⟷ TRDb, OSL1 ⟷ OSL1b,
    OSL2 ⟷ OSL2b, BRG ⟷ BRGb, STV ⟷ STVb}];
network = GraphUnion[Primary, Secondary, ConnectionPS, VertexLabels → "Name"];
```

A datacenter node connects to TRD and TRDb nodes.

```
datacenter1 = Graph[{DC1 ⟷ TRD, DC1 ⟷ TRDb}, VertexLabels → "Name"];
```

```
vnf1 = Graph[{VNF1 ⟷ DC1}, VertexLabels → "Name"];
```

```
vnf2 = Graph[{VNF2 ⟷ DC1}, VertexLabels → "Name"];
```

```
vnf3 = Graph[{VNF3 ⟷ DC1}, VertexLabels → "Name"];
```

A NFV - MANO node connects to TRD and TRDb nodes.

```
mano = Graph[{MANO1 ⟷ BRG, MANO1 ⟷ BRGb}, VertexLabels → "Name"];
```

```
Datacenter = GraphUnion[network, datacenter1, VertexShapeFunction → "Name"];
```

```
NFV = GraphUnion[network, datacenter1,
   vnf1, vnf2, vnf3, mano, VertexShapeFunction → "Name"];
```

Connect to endpoints.

```
endpoint1 = Graph[{O1 → TRD, O1 → TRDb}, VertexLabels → "Name"];
```

```
endpoint2 = Graph[{STV → D1, STVb → D1}, VertexLabels → "Name"];
```

```
MANON =
  GraphUnion[network, mano, endpoint1, endpoint2, VertexShapeFunction → "Name"];
```

Network with endpoints only.

```
endpointsnetwork = GraphUnion[Datacenter, endpoint1, endpoint2];
```

```
HighlightGraph[endpointsnetwork,
  {network, endpoint1, endpoint2, datacenter1}, VertexLabels → "Name"];
```

Network with endpoints and NFV elements.

```
overallNetwork = GraphUnion[NFV, endpoint1, endpoint2];
```

```
HighlightGraph[overallNetwork,
  {network, mano, vnf1, vnf2, vnf3, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Find the path

Endpoints

```
Org[i_] := ToExpression["O" <> ToString[i]];
```

```
Dest[i_] := ToExpression["D" <> ToString[i]];
```

Datacenters

```
box[i_] := ToExpression["DC" <> ToString[i]];
```

VNFs

```
Vir[i_] := ToExpression["VNF" <> ToString[i]];
```

MANO

```
Ctrl[i_] := ToExpression["MANO" <> ToString[i]];
```

Paths between endpoints(service provider) to endpoints(user) include the service function chain.

```
Cpathsf[i_, j_] := FindPath[overallNetwork, Vir[1], Dest[j], Infinity, All];
```

```
Cpathss[i_, j_] := FindPath[overallNetwork, Vir[2], Vir[1], Infinity, All];
```

```
Cpathst[i_, j_] := FindPath[overallNetwork, Vir[3], Vir[2], Infinity, All];
```

```
Cpathse[i_, j_] := FindPath[overallNetwork, Org[i], Vir[3], Infinity, All];
```

Paths between endpoints and MANO

```
Spaths[i_, j_] := FindPath[MANON, Org[i], Ctrl[j], Infinity, All];
```

```
Upaths[i_, j_] := FindPath[MANON, Ctrl[i], Dest[j], Infinity, All];
```

Paths between VNFs and MANO

```
Vpaths[i_, j_] := FindPath[NFV, Vir[i], Ctrl[j], Infinity, All];
```

Define parameters

```
NoOfPoP = 1; NoOfVNF = 3; NoOfMANO = 1;
```

```
CpathsfM = Table[Cpathsf[i, j], {i, 1}, {j, 1, NoOfPoP}];
```

```
CpathssM = Table[Cpathss[i, j], {i, 1}, {j, 1}];
```

```
CpathstM = Table[Cpathst[i, j], {i, 1}, {j, 1}];
```

```
CpathseM = Table[Cpathse[i, j], {i, 1}, {j, 1}];
```

```
SpathsM = Table[Spaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
UpathsM = Table[Upaths[i, j], {i, 1, NoOfMANO}, {j, 1, NoOfPoP}];
```

```
VpathsM = Table[Vpaths[i, j], {i, 1, NoOfVNF}, {j, 1, NoOfMANO}];
```

Converting paths into network elements

If both nodes and links may fail, we have to include both in the list of elements, since we cannot maintain the sequence in compond paths.NB : The traffic sources and the related links to/from them must not be included in the network elements.

Mapping Vertexes (in order to valuate unoriented links)

```
VMap = {TRD, TRDb, OSL1, OSL1b, OSL2,
    OSL2b, STV, STVb, BRG, BRGb, DC1, MANO1, VNF1, VNF2, VNF3};
```

Service chain .

VNF1 and user endpoint.

```
fnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathsfM[[i, j, k]], 1], -1],
      {k, Length[CpathsfM[[i, j]]]}]), {2}];
```

```
flinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathsfM[[i, j, k]], -1], Drop[
          CpathsfM[[i, j, k]], 1]}], -1], {k, Length[CpathsfM[[i, j]]]}]), {2}];
```

```
fnodesinpathsM = Table[fnodesinpaths[i, j], {i, 1}, {j, 1}];
```

```
flinksinpathM = Table[flinksinpath[i, j], {i, 1}, {j, 1}];
```

```
Felements[i_, j_] := Join[fnodesinpathsM[[i, j]], flinksinpathM[[i, j]], 2];
```

```
FelementsM = Table[Felements[i, j], {i, 1}, {j, 1}];
```

VNF2 and VNF1

```
snodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathssM[[i, j, k]], 1], -1],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
```

```
slinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathssM[[i, j, k]], -1], Drop[CpathssM[[i, j, k]], 1]}],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
```

```
snodesinpathsM = Table[snodesinpaths[i, j], {i, 1}, {j, 1}];
```

```
slinksinpathM = Table[slinksinpath[i, j], {i, 1}, {j, 1}];
```

```
Selements[i_, j_] := Join[snodesinpathsM[[i, j]], slinksinpathM[[i, j]], 2];
```

```
SelementsM = Table[Selements[i, j], {i, 1}, {j, 1}];
```

VNF3 and VNF2

```
tnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathstM[[i, j, k]], 1], -1],
      {k, Length[CpathstM[[i, j]]]}]), {2}];
```

```
tlinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathstM[[i, j, k]], -1], Drop[CpathstM[[i, j, k]], 1]}],
      {k, Length[CpathstM[[i, j]]]}]), {2}];
```

```
tnodesinpathsM = Table[tnodesinpaths[i, j], {i, 1}, {j, 1}];
```

```
tlinksinpathM = Table[tlinksinpath[i, j], {i, 1}, {j, 1}];

Telements[i_, j_] := Join[tnodesinpathsM[[i, j]], tlinksinpathM[[i, j]], 2];

TelementsM = Table[Telements[i, j], {i, 1}, {j, 1}];
```

Service provider endpoints and VNF3

```
enodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathseM[[i, j, k]], 1], -1],
      {k, Length[CpathseM[[i, j]]]}]), {2}];

elinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathseM[[i, j, k]], -1], Drop[
          CpathseM[[i, j, k]], 1]}], 1], {k, Length[CpathseM[[i, j]]]}]), {2}];

enodesinpathsM = Table[enodesinpaths[i, j], {i, 1}, {j, 1}];

elinksinpathM = Table[elinksinpath[i, j], {i, 1}, {j, 1}];

Eelements[i_, j_] := Join[enodesinpathsM[[i, j]], elinksinpathM[[i, j]], 2];

EelementsM = Table[Eelements[i, j], {i, 1}, {j, 1}];
```

Endpoints connect to MANO

Service provider endpoint and MANO

```
dmanonodesinpaths[i_, j_] := Map[n# &,
    (Table[Drop[SpathsM[[i, j, k]], 1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[SpathsM[[i, j, k]], -1],
         Drop[SpathsM[[i, j, k]], 1]}], 1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanonodesinpathsM = Table[dmanonodesinpaths[i, j], {i, 1}, {j, 1}];

dmanolinksinpathM = Table[dmanolinksinpath[i, j], {i, 1}, {j, 1}];

Dmanoelements[i_, j_] :=
  Join[dmanonodesinpathsM[[i, j]], dmanolinksinpathM [[i, j]], 2];

DmanoelementsM = Table[Dmanoelements[i, j], {i, 1}, {j, 1}];
```

User endpoint and MANO

```
omanonodesinpaths[i_, j_] := Map[n# &,
    (Table[Drop[UpathsM[[i, j, k]], -1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[UpathsM[[i, j, k]], -1],
         Drop[UpathsM[[i, j, k]], 1]}], -1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanonodesinpathsM = Table[omanonodesinpaths[i, j], {i, 1}, {j, 1}];

omanolinksinpathM = Table[omanolinksinpath[i, j], {i, 1}, {j, 1}];
```

```
Omanoelements[i_, j_] :=
  Join[omanonodesinpathsM[[i, j]], omanolinksinpathM [[i, j]], 2];

OmanoelementsM = Table[Omanoelements[i, j], {i, 1}, {j, 1}];
```

VNFS connect to MANO

```
nodesin[i_, j_] := Map[n# &,
   (Table[Drop[VpathsM[[i, j, k]], 1], {k, Length[VpathsM[[i, j]]]}]), {2}];

linksin[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
   (Table[Transpose[{Drop[VpathsM[[i, j, k]], -1], Drop[VpathsM[[i, j, k]], 1]}],
     {k, Length[VpathsM[[i, j]]]}]), {2}];

nodesinM = Table[nodesin[i, j], {i, 1, NoOfVNF}, {j, 1, NoOfMANO}];

linksinM = Table[linksin[i, j], {i, 1, NoOfVNF}, {j, 1, NoOfMANO}];

Velements[i_, j_] := Join[nodesinM[[i, j]], linksinM[[i, j]], 2];

VelementsM =
  Table[If[i != j, Velements[i, j]], {i, 1, NoOfVNF}, {j, 1, NoOfMANO}];
```

The structure function and its analysis

convers the sets of elements providing a working path and control to a Boolean expression for reduction & analysis.

Service chain .

VNF1 and user endpoint.

```
Φf[i_, j_] := Table[{FelementsM[[i, j, k]] /. List → And},
   {k, Length[FelementsM[[i, j]]]}] /. List → Or

ΦfM = Table[Φf[i, j], {i, 1}, {j, 1}];

ΦfT = Table[{ΦfM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF2 and VNF1

```
Φs[i_, j_] := Table[{SelementsM[[i, j, k]] /. List → And},
   {k, Length[SelementsM[[i, j]]]}] /. List → Or

ΦsM = Table[Φs[i, j], {i, 1}, {j, 1}];

ΦsT = Table[{ΦsM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF3 and VNF2

```
Φt[i_, j_] := Table[{TelementsM[[i, j, k]] /. List → And},
   {k, Length[TelementsM[[i, j]]]}] /. List → Or

ΦtM = Table[Φt[i, j], {i, 1}, {j, 1}];
```

```
ΦtT = Table[{ΦtM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF3 and endpoint service provider

```
Φe[i_, j_] := Table[{EelementsM[[i, j, k]] /. List → And},
   {k, Length[EelementsM[[i, j]]]}] /. List → Or
```

```
ΦeM = Table[Φt[i, j], {i, 1}, {j, 1}];
```

```
ΦeT = Table[{ΦeM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

Endpoints connect to MANO

MANO and endpoint service provider

```
Φd[i_, j_] := Table[{DmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[DmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦdM = Table[Φd[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
ΦdT = Table[{ΦdM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

MANO and endpoint service provider

```
Φo[i_, j_] := Table[{OmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[OmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦoM = Table[Φo[i, j], {i, 1, NoOfMANO}, {j, 1, NoOfPoP}];
```

```
ΦoT = Table[{ΦoM[[k]] /. List → Or}, {k, NoOfMANO}] /. List → And;
```

MANO-> VNFs

```
Φv[i_, j_] := Table[{VelementsM[[i, j, k]] /. List → And},
   {k, Length[VelementsM[[i, j]]]}] /. List → Or
```

```
ΦvM = Table[Φv[i, j], {i, 1, NoOfVNF}, {j, 1, NoOfMANO}];
```

```
ΦvT = And @@ Flatten[Diagonal[ΦvM, #] & /@ Range[NoOfVNF - 1]];
```

```
Φnfv = ΦfT && ΦsT && ΦeT && ΦtT && ΦvT && ΦdT && ΦoT // Simplify;
```

Extracting the minimum paths and cut

The minimum path and cut sets are extracted from the above logical expressions by replacing the heads of the expressions. The And and Or head are replaced by List heads. The not operation are replaced by identity and removed. Be aware that the operation may twist the meaning unless the logical expand produces results on standard form.

Functions

```
MinPaths[s_] := BooleanConvert[s] /. {And -> List, Or -> List}
```

```
MinCuts[s_] :=
 BooleanConvert[s, "CNF" ] /. {And -> List, Or -> List, Not -> Identity}
```

NFV

```
NFVminpath = MinPaths[Φnfv];
```

```
NFVmincut = MinCuts[Φnfv];
```

```
Export["D://NTNU 2 year//Master thesis//structure
   analysis//data//NFVmincut_norway_1e.mx", NFVmincut, "MX"]
```

```
D://NTNU 2 year//Master
  thesis//structure analysis//data//NFVmincut_norway_1e.mx
```

Cardinality cut sets.

```
Scenario1 = Length /@ (If[Head[#] =!= List, {#}, #] & /@ NFVmincut);
```

```
BinCounts[%, {1, Max[%] + 1, 1}]
```

```
{5, 9, 8, 59, 232, 442, 446, 320, 195, 57, 12}
```

```
Position[Scenario1, _?(# < 2 &)]
```

```
{{600}, {601}, {1709}, {1710}, {1711}}
```

```
Position[Scenario1, _?(1 < # < 4 &)]
```

```
{{1}, {214}, {229}, {251}, {253}, {298}, {483}, {506}, {542},
 {552}, {595}, {1307}, {1515}, {1562}, {1613}, {1708}, {1785}}
```

Appendix B

Structural analysis script of Scenario 1 in Mathematica

Structure model of Scenario I

Define network

```
Primary = Graph[{TRD ⟷ OSL1, TRD ⟷ OSL2,
    TRD ⟷ BRG, TRD ⟷ STV, OSL1 ⟷ OSL2, OSL1 ⟷ STV, STV ⟷ BRG}];
Secondary = Graph[{TRDb ⟷ OSL2b, TRDb ⟷ BRGb, TRDb ⟷ STVb,
    OSL1b ⟷ OSL2b, OSL1b ⟷ STVb, STVb ⟷ BRGb}];
ConnectionPS = Graph[{TRD ⟷ TRDb, OSL1 ⟷ OSL1b,
    OSL2 ⟷ OSL2b, BRG ⟷ BRGb, STV ⟷ STVb}];
network = GraphUnion[Primary, Secondary, ConnectionPS, VertexLabels → "Name"];
```

A datacenter node connects to TRD and TRDb nodes.
A datacenter node connects to OSL2,OSL1,OSL1B,OSL2B nodes.

```
datacenter1 = Graph[{DC1 ⟷ TRD, DC1 ⟷ TRDb}, VertexLabels → "Name"];

datacenter2 = Graph[
   {OSL2 ⟷ DC2, OSL1 ⟷ DC2, OSL1b ⟷ DC2, OSL2b ⟷ DC2}, VertexLabels → "Name"];

vnf1 = Graph[{VNF1 ⟷ DC1}, VertexLabels → "Name"];

vnf2 = Graph[{VNF2 ⟷ DC1}, VertexLabels → "Name"];

vnf3 = Graph[{VNF3 ⟷ DC2}, VertexLabels → "Name"];
```

A NFV-MANO node connects to BRG,BRGb nodes.

```
mano = Graph[{MANO1 ⟷ BRG, MANO1 ⟷ BRGb}, VertexLabels → "Name"];


Datacenter =
  GraphUnion[network, datacenter1, datacenter2, VertexShapeFunction → "Name"];


NFV = GraphUnion[network, datacenter1, datacenter2,
   vnf1, vnf2, vnf3, mano, VertexShapeFunction → "Name"];

NFV1 = GraphUnion[network, datacenter1, vnf1, VertexShapeFunction → "Name"];

NFV11 =
  GraphUnion[network, datacenter1, vnf2, vnf1, VertexShapeFunction → "Name"];

NFV2 = GraphUnion[network, datacenter1, vnf1,
   vnf2, datacenter2, vnf3, VertexShapeFunction → "Name"];

NFV3 = GraphUnion[network, datacenter2, vnf3, VertexShapeFunction → "Name"];

MANON = GraphUnion[network, mano, VertexShapeFunction → "Name"];

MANON1 = GraphUnion[network, mano,
   datacenter1, vnf1, vnf2, VertexShapeFunction → "Name"];
MANON2 = GraphUnion[network, mano, datacenter2, vnf3,
   VertexShapeFunction → "Name"];
```

Connect to endpoints.

```
endpoint1 = Graph[{O1 → TRD, O1 → TRDb}, VertexLabels → "Name"];
```

```
endpoint2 = Graph[{STV → D1, STVb → D1}, VertexLabels → "Name"];
```

Network with endpoints only.

```
endpointsnetwork3 = GraphUnion[MANON, endpoint1, endpoint2];
```

```
endpointsnetwork1 = GraphUnion[NFV1, endpoint2];
```

```
endpointsnetwork2 = GraphUnion[NFV3, endpoint1];
```

```
endpointsnetwork = GraphUnion[Datacenter, endpoint1, endpoint2];
```

```
HighlightGraph[endpointsnetwork,
  {network, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Network with endpoints and NFV elements.

```
overallNetwork = GraphUnion[NFV, endpoint1, endpoint2];
```

```
HighlightGraph[overallNetwork,
  {network, mano, vnf1, vnf2, vnf3, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Find the path

Endpoints

```
Org[i_] := ToExpression["O" <> ToString[i]];
```

```
Dest[i_] := ToExpression["D" <> ToString[i]];
```

Datacenters

```
box[i_] := ToExpression["DC" <> ToString[i]];
```

VNFs

```
Vir[i_] := ToExpression["VNF" <> ToString[i]];
```

MANO

```
Ctrl[i_] := ToExpression["MANO" <> ToString[i]];
```

Paths between endpoints(service provider) to endpoints(user) include the service function chain.

```
Cpathsf[i_, j_] := FindPath[endpointsnetwork1, Vir[1], Dest[1], Infinity, All]
```

```
Cpathss[i_, j_] := FindPath[NFV11, Vir[2], Vir[1], Infinity, All]
```

```
Cpathst[i_, j_] := FindPath[NFV2, Vir[3], Vir[2], Infinity, All]
```

```
Cpathse[i_, j_] := FindPath[endpointsnetwork2, Org[1], Vir[3], Infinity, All]
```

Paths between endpoints and MANO

```
Spaths[i_, j_] := FindPath[endpointsnetwork3, Org[i], Ctrl[j], Infinity, All];
```

```
Upaths[i_, j_] := FindPath[endpointsnetwork3, Ctrl[i], Dest[j], Infinity, All];
```

Paths between VNFs and MANO

```
Vpathsf[i_, j_] := FindPath[MANON1, Vir[1], Ctrl[j], Infinity, All];

Vpathss[i_, j_] := FindPath[MANON1, Vir[2], Ctrl[j], Infinity, All];

Vpathst[i_, j_] := FindPath[MANON2, Vir[3], Ctrl[j], Infinity, All];
```

Define parameters.

```
NoOfPoP = 1;  NoOfVNF = 3; NoOfMANO = 1;
```

```
CpathsfM = Table[Cpathsf[i, j], {i, 1}, {j, 1}];

CpathssM = Table[Cpathss[i, j], {i, 1}, {j, 1}];

CpathstM = Table[Cpathst[i, j], {i, 1}, {j, 1}];

CpathseM = Table[Cpathse[i, j], {i, 1}, {j, 1}];

SpathsM = Table[Spaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];

UpathsM = Table[Upaths[i, j], {i, 1, NoOfMANO}, {j, 1, NoOfPoP}];

VpathsfM = Table[Vpathsf[i, j], {i, 1}, {j, 1}];

VpathssM = Table[Vpathss[i, j], {i, 1}, {j, 1}];

VpathstM = Table[Vpathst[i, j], {i, 1}, {j, 1}];
```

Converting paths into network elements

If both nodes and links may fail, we have to include both in the list of elements, since we cannot maintain the sequence in compond paths.NB : The traffic sources and the related links to/from them must not be included in the network elements.

Mapping Vertexes (in order to valuate unoriented links)

```
VMap = {TRD, TRDb, OSL1, OSL1b, OSL2, OSL2b,
    STV, STVb, BRG, BRGb, DC1, DC2, MANO1, VNF1, VNF2, VNF3};
```

Service chain .

VNF1 and user endpoint.

```
fnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathsfM[[i, j, k]], 1], -1],
      {k, Length[CpathsfM[[i, j]]]}]), {2}];
```

```
flinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathsfM[[i, j, k]], -1], Drop[
          CpathsfM[[i, j, k]], 1]}], -1], {k, Length[CpathsfM[[i, j]]]}]), {2}];
```

```
fnodesinpathsM = Table[fnodesinpaths[i, j], {i, 1}, {j, 1}];
```

```
flinksinpathM = Table[flinksinpath[i, j], {i, 1}, {j, 1}];
```

```
Felements[i_, j_] := Join[fnodesinpathsM[[i, j]], flinksinpathM[[i, j]], 2];

FelementsM = Table[Felements[i, j], {i, 1}, {j, 1}];
```

VNF2 and VNF1

```
snodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathssM[[i, j, k]], 1], -1],
      {k, Length[CpathssM[[i, j]]]}]), {2}];

slinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathssM[[i, j, k]], -1], Drop[CpathssM[[i, j, k]], 1]}],
      {k, Length[CpathssM[[i, j]]]}]), {2}];

snodesinpathsM = Table[snodesinpaths[i, j], {i, 1}, {j, 1}];

slinksinpathM = Table[slinksinpath[i, j], {i, 1}, {j, 1}];

Selements[i_, j_] := Join[snodesinpathsM[[i, j]], slinksinpathM[[i, j]], 2];

SelementsM = Table[Selements[i, j], {i, 1}, {j, 1}];
```

VNF3 and VNF2

```
tnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathstM[[i, j, k]], 1], -1],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tlinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathstM[[i, j, k]], -1], Drop[CpathstM[[i, j, k]], 1]}],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tnodesinpathsM = Table[tnodesinpaths[i, j], {i, 1}, {j, 1}];

tlinksinpathM = Table[tlinksinpath[i, j], {i, 1}, {j, 1}];

Telements[i_, j_] := Join[tnodesinpathsM[[i, j]], tlinksinpathM[[i, j]], 2];

TelementsM = Table[Telements[i, j], {i, 1}, {j, 1}];
```

Service provider endpoints and VNF3

```
enodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathseM[[i, j, k]], 1], -1],
      {k, Length[CpathseM[[i, j]]]}]), {2}];

elinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathseM[[i, j, k]], -1], Drop[
          CpathseM[[i, j, k]], 1]}], 1], {k, Length[CpathseM[[i, j]]]}]), {2}];

enodesinpathsM = Table[enodesinpaths[i, j], {i, 1}, {j, 1}];

elinksinpathM = Table[elinksinpath[i, j], {i, 1}, {j, 1}];

Eelements[i_, j_] := Join[enodesinpathsM[[i, j]], elinksinpathM[[i, j]], 2];

EelementsM = Table[Eelements[i, j], {i, 1}, {j, 1}];
```

Endpoints connect to MANO

Service provider endpoint and MANO

```
dmanonodesinpaths[i_, j_] := Map[n# &,
    (Table[Drop[SpathsM[[i, j, k]], 1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[SpathsM[[i, j, k]], -1],
        Drop[SpathsM[[i, j, k]], 1]}], 1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanonodesinpathsM = Table[dmanonodesinpaths[i, j], {i, 1}, {j, 1}];

dmanolinksinpathM = Table[dmanolinksinpath[i, j], {i, 1}, {j, 1}];

Dmanoelements[i_, j_] :=
  Join[dmanonodesinpathsM[[i, j]], dmanolinksinpathM [[i, j]], 2];

DmanoelementsM = Table[Dmanoelements[i, j], {i, 1}, {j, 1}];
```

User endpoint and MANO

```
omanonodesinpaths[i_, j_] := Map[n# &,
    (Table[Drop[UpathsM[[i, j, k]], -1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[UpathsM[[i, j, k]], -1],
        Drop[UpathsM[[i, j, k]], 1]}], -1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanonodesinpathsM = Table[omanonodesinpaths[i, j], {i, 1}, {j, 1}];

omanolinksinpathM = Table[omanolinksinpath[i, j], {i, 1}, {j, 1}];

Omanoelements[i_, j_] :=
  Join[omanonodesinpathsM[[i, j]], omanolinksinpathM [[i, j]], 2];

OmanoelementsM = Table[Omanoelements[i, j], {i, 1}, {j, 1}];
```

VNFS connect to MANO

VNF1 TO MANO

```
vnff[i_, j_] := Map[n# &,
    (Table[Drop[VpathsfM[[i, j, k]], 1], {k, Length[VpathsfM[[i, j]]]}]), {2}];

linksinf[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathsfM[[i, j, k]], -1], Drop[VpathsfM[[i, j, k]], 1]}],
     {k, Length[VpathsfM[[i, j]]]}]), {2}];

vnffM = Table[vnff[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksinfM = Table[linksinf[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementsf[i_, j_] := Join[vnffM[[i, j]], linksinfM[[i, j]], 2];
```

```
VelementsfM = Table[Velementsf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF2 TO MANO

```
vnfs[i_, j_] := Map[n# &,
    (Table[Drop[VpathssM[[i, j, k]], 1], {k, Length[VpathssM[[i, j]]]}]), {2}];
```

```
linksins[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathssM[[i, j, k]], -1], Drop[VpathssM[[i, j, k]], 1]}],
     {k, Length[VpathssM[[i, j]]]}]), {2}];
```

```
vnfsM = Table[vnfs[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];
```

```
linksinsM = Table[linksins[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
Velementss[i_, j_] := Join[vnfsM[[i, j]], linksinsM[[i, j]], 2];
```

```
VelementssM = Table[Velementss[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF3 TO MANO

```
vnft[i_, j_] := Map[n# &,
    (Table[Drop[VpathstM[[i, j, k]], 1], {k, Length[VpathstM[[i, j]]]}]), {2}];
```

```
linksint[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathstM[[i, j, k]], -1], Drop[VpathstM[[i, j, k]], 1]}],
     {k, Length[VpathstM[[i, j]]]}]), {2}];
```

```
vnftM = Table[vnft[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];
```

```
linksintM = Table[linksint[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
Velementst[i_, j_] := Join[vnftM[[i, j]], linksintM[[i, j]], 2];
```

```
VelementstM = Table[Velementst[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

The structure function and its analysis

convers the sets of elements providing a working path and control to a Boolean expression for reduction & analysis.

Service chain .

VNF1 and user endpoint.

```
Φf[i_, j_] := Table[{FelementsM[[i, j, k]] /. List → And},
    {k, Length[FelementsM[[i, j]]]}] /. List → Or
```

```
ΦfM = Table[Φf[i, j], {i, 1}, {j, 1}];
```

```
ΦfT = Table[{ΦfM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF2 and VNF1

```
Φs[i_, j_] := Table[{SelementsM[[i, j, k]] /. List → And},
   {k, Length[SelementsM[[i, j]]]}] /. List → Or
```

```
ΦsM = Table[Φs[i, j], {i, 1}, {j, 1}];
```

```
ΦsT = Table[{ΦsM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF3 and VNF2

```
Φt[i_, j_] := Table[{TelementsM[[i, j, k]] /. List → And},
   {k, Length[TelementsM[[i, j]]]}] /. List → Or
```

```
ΦtM = Table[Φt[i, j], {i, 1}, {j, 1}];
```

```
ΦtT = Table[{ΦtM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF3 and endpoint service provider

```
Φe[i_, j_] := Table[{EelementsM[[i, j, k]] /. List → And},
   {k, Length[EelementsM[[i, j]]]}] /. List → Or
```

```
ΦeM = Table[Φt[i, j], {i, 1}, {j, 1}];
```

```
ΦeT = Table[{ΦeM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

Endpoints connect to MANO

MANO and endpoint service provider

```
Φd[i_, j_] := Table[{DmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[DmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦdM = Table[Φd[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦdT = Table[{ΦdM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

MANO and endpoint service provider

```
Φo[i_, j_] := Table[{OmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[OmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦoM = Table[Φo[i, j], {i, 1, NoOfMANO}, {j, 1}];
```

```
ΦoT = Table[{ΦoM[[k]] /. List → Or}, {k, NoOfMANO}] /. List → And;
```

VNFs-> MANO

VNF1 TO MANO

```
Φvf[i_, j_] := Table[{VelementsfM[[i, j, k]] /. List → And},
   {k, Length[VelementsfM[[i, j]]]}] /. List → Or
```

```
ΦvfM = Table[Φvf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvfT = Table[{ΦvfM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF2 TO MANO

```
Φvs[i_, j_] := Table[{VelementssM[[i, j, k]] /. List → And},
   {k, Length[VelementssM[[i, j]]]}] /. List → Or
```

```
ΦvsM = Table[Φvs[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvsT = Table[{ΦvsM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF3 TO MANO

```
Φvt[i_, j_] := Table[{VelementstM[[i, j, k]] /. List → And},
   {k, Length[VelementstM[[i, j]]]}] /. List → Or
```

```
ΦvtM = Table[Φvt[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvtT = Table[{ΦvtM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

```
Φnfv = ΦfT && ΦsT && ΦeT && ΦtT && ΦvfT && ΦdT && ΦoT && ΦvsT && ΦvtT // Simplify;
```

Extracting the minimum paths and cut

The minimum path and cut sets are extracted from the above logical expressions by replacing the heads of the expressions. The And and Or head are replaced by List heads. The not operation are replaced by identity and removed. Be aware that the operation may twist the meaning unless the logical expand produces results on standard form.

Functions

```
MinPaths[s_] := BooleanConvert[s] /. {And -> List, Or -> List}
```

```
MinCuts[s_] :=
 BooleanConvert[s, "CNF" ] /. {And -> List, Or -> List, Not -> Identity}
```

NFV

```
NFVminpath = MinPaths[Φnfv];
```

```
NFVmincut = MinCuts[Φnfv];
```

```
Export["D://NTNU 2 year//Master thesis//structure
   analysis//data//NFVmincut_norway_2e.mx", NFVmincut, "MX"]
```

```
D://NTNU 2 year//Master
  thesis//structure analysis//data//NFVmincut_norway_2e.mx
```

Cardinality cut sets.

```
Scenario2 = Length /@ (If[Head[#] =!= List, {#}, #] & /@ NFVmincut);
```

```
BinCounts[Scenario2, {1, Max[%] + 1, 1}]
```

```
{6, 9, 8, 120, 360, 639, 687, 500, 281, 112, 49, 6, 6}
```

```
Position[Scenario2, _?(2 < # < 4 &)]
```

```
{{303}, {318}, {346}, {426}, {685}, {708}, {749}, {839}}
```

Appendix C

Structural analysis script of Scenario 3 in Mathematica

Structure model of Scenario 2

Define network

```
Primary = Graph[{TRD ⟷ OSL1, TRD ⟷ OSL2,
     TRD ⟷ BRG, TRD ⟷ STV, OSL1 ⟷ OSL2, OSL1 ⟷ STV, STV ⟷ BRG}];
Secondary = Graph[{TRDb ⟷ OSL2b, TRDb ⟷ BRGb, TRDb ⟷ STVb,
     OSL1b ⟷ OSL2b, OSL1b ⟷ STVb, STVb ⟷ BRGb}];
ConnectionPS = Graph[{TRD ⟷ TRDb, OSL1 ⟷ OSL1b,
     OSL2 ⟷ OSL2b, BRG ⟷ BRGb, STV ⟷ STVb}];
network = GraphUnion[Primary, Secondary, ConnectionPS, VertexLabels → "Name"];


datacenter1 = Graph[{DC1 ⟷ TRD, DC1 ⟷ TRDb}, VertexLabels → "Name"];
mano = Graph[{MANO1 ⟷ BRG, MANO1 ⟷ BRGb}, VertexLabels → "Name"];
datacenter2 = Graph[
    {OSL2 ⟷ DC2, OSL1 ⟷ DC2, OSL1b ⟷ DC2, OSL2b ⟷ DC2}, VertexLabels → "Name"];

vnf1 = Graph[{VNF1 ⟷ DC1}, VertexLabels → "Name"];

vnf2 = Graph[{VNF2 ⟷ DC1, VNF2 ⟷ DC2}, VertexLabels → "Name"];

vnf3 = Graph[{VNF3 ⟷ DC2}, VertexLabels → "Name"];

Datacenter =
  GraphUnion[network, datacenter1, datacenter2, VertexShapeFunction → "Name"];

NFV = GraphUnion[network, datacenter1, datacenter2,
    vnf1, vnf2, vnf3, mano, VertexShapeFunction → "Name"];
```

Connect to endpoints.

```
endpoint1 = Graph[{O1 → TRD, O1 → TRDb}, VertexLabels → "Name"];

endpoint2 = Graph[{STV → D1, STVb → D1}, VertexLabels → "Name"];
```

Network with endpoints only.

```
endpointsnetwork1 =
  GraphUnion[vnf1, datacenter1, network, endpoint2, VertexLabels → "Name"];

endpointsnetwork2 = GraphUnion[datacenter2,
    vnf2, vnf1, datacenter1, network, VertexLabels → "Name"];

endpointsnetwork3 = GraphUnion[datacenter2,
    vnf2, vnf3, datacenter1, network, VertexLabels → "Name"];

endpointsnetwork4 =
  GraphUnion[datacenter2, vnf3, endpoint1, network, VertexLabels → "Name"];

endpointsnetwork5 = GraphUnion[mano, endpoint1, network, VertexLabels → "Name"];

endpointsnetwork6 = GraphUnion[mano, endpoint2, network, VertexLabels → "Name"];

NFV1 = GraphUnion[network, datacenter1, vnf1, mano, VertexShapeFunction → "Name"];

NFV2 = GraphUnion[network, datacenter1,
    datacenter2, vnf2, mano, VertexShapeFunction → "Name"];
```

```
NFV3 = GraphUnion[network, datacenter2, vnf3, mano, VertexShapeFunction → "Name"];
```

```
HighlightGraph[endpointsnetwork1,
  {network, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Network with endpoints and NFV elements.

```
overallNetwork = GraphUnion[NFV, endpoint1, endpoint2];
```

```
HighlightGraph[overallNetwork,
  {network, mano, vnf1, vnf2, vnf3, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Find the path

Endpoints

```
Org[i_] := ToExpression["O" <> ToString[i]];
```

```
Dest[i_] := ToExpression["D" <> ToString[i]];
```

Datacenters

```
box[i_] := ToExpression["DC" <> ToString[i]];
```

VNFs

```
Vir[i_] := ToExpression["VNF" <> ToString[i]];
```

MANO

```
Ctrl[i_] := ToExpression["MANO" <> ToString[i]];
```

Paths between endpoints(service provider) to endpoints(user) include the service function chain.

```
Cpathsf[i_, j_] := FindPath[endpointsnetwork1, Vir[1], Dest[j], Infinity, All]
```

```
Cpathss[i_, j_] := FindPath[endpointsnetwork2, Vir[2], Vir[1], Infinity, All]
```

```
Cpathst[i_, j_] := FindPath[endpointsnetwork3, Vir[3], Vir[2], Infinity, All]
```

```
Cpathse[i_, j_] := FindPath[endpointsnetwork4, Org[i], Vir[3], Infinity, All]
```

Paths between endpoints and MANO

```
Spaths[i_, j_] := FindPath[endpointsnetwork5, Org[i], Ctrl[j], Infinity, All];
```

```
Upaths[i_, j_] := FindPath[endpointsnetwork6, Ctrl[i], Dest[j], Infinity, All];
```

Paths between VNFs and MANO

```
Vpathsf[i_, j_] := FindPath[NFV1, Vir[1], Ctrl[j], Infinity, All];
```

```
Vpathss[i_, j_] := FindPath[NFV2, Vir[2], Ctrl[j], Infinity, All];
```

```
Vpathst[i_, j_] := FindPath[NFV3, Vir[3], Ctrl[j], Infinity, All];
```

Define parameters.

```
NoOfPoP = 1;  NoOfVNF = 3; NoOfMANO = 1;
```

```
CpathsfM = Table[Cpathsf[i, j], {i, 1}, {j, 1}];
```

```
CpathssM = Table[Cpathss[i, j], {i, 1}, {j, 1}];
```

```
CpathstM = Table[Cpathst[i, j], {i, 1}, {j, 1}];

CpathseM = Table[Cpathse[i, j], {i, 1}, {j, 1}];

SpathsM = Table[Spaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];

UpathsM = Table[Upaths[i, j], {i, 1, NoOfMANO}, {j, 1, NoOfPoP}];

VpathsfM = Table[Vpathsf[i, j], {i, 1}, {j, 1, NoOfMANO}];

VpathssM = Table[Vpathss[i, j], {i, 1}, {j, 1, NoOfMANO}];

VpathstM = Table[Vpathst[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

Converting paths into network elements

If both nodes and links may fail, we have to include both in the list of elements, since we cannot maintain the sequence in compond paths.NB : The traffic sources and the related links to/from them must not be included in the network elements

Mapping Vertexes (in order to valuate unoriented links)

```
VMap = {TRD, TRDb, OSL1, OSL1b, OSL2, OSL2b,
   STV, STVb, BRG, BRGb, DC1, DC2, DC3, MANO1, VNF1, VNF2, VNF3};
```

Service chain .

VNF1 and user endpoint.

```
fnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathsfM[[i, j, k]], 1], -1],
      {k, Length[CpathsfM[[i, j]]]}]), {2}];

flinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathsfM[[i, j, k]], -1], Drop[
          CpathsfM[[i, j, k]], 1]}], -1], {k, Length[CpathsfM[[i, j]]]}]), {2}];

fnodesinpathsM = Table[fnodesinpaths[i, j], {i, 1}, {j, 1}];

flinksinpathM = Table[flinksinpath[i, j], {i, 1}, {j, 1}];

Felements[i_, j_] := Join[fnodesinpathsM[[i, j]], flinksinpathM[[i, j]], 2];

FelementsM = Table[Felements[i, j], {i, 1}, {j, 1}];
```

VNF2 and VNF1

```
snodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathssM[[i, j, k]], 1], -1],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
slinksinpath[i_, j_] := Map[If[Position[VMap, First[#]][[1, 1]] <
       Position[VMap, Last[#]][[1, 1]], Subscript[l, First[#], Last[#]],
      Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathssM[[i, j, k]], -1], Drop[CpathssM[[i, j, k]], 1]}],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
snodesinpathsM = Table[snodesinpaths[i, j], {i, 1}, {j, 1}];

slinksinpathM = Table[slinksinpath[i, j], {i, 1}, {j, 1}];

Selements[i_, j_] := Join[snodesinpathsM[[i, j]], slinksinpathM[[i, j]], 2];
SelementsM = Table[Selements[i, j], {i, 1}, {j, 1}];
```

VNF3 and VNF2

```
tnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathstM[[i, j, k]], 1], -1],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tlinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathstM[[i, j, k]], -1], Drop[CpathstM[[i, j, k]], 1]}],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tnodesinpathsM = Table[tnodesinpaths[i, j], {i, 1}, {j, 1}];

tlinksinpathM = Table[tlinksinpath[i, j], {i, 1}, {j, 1}];

Telements[i_, j_] := Join[tnodesinpathsM[[i, j]], tlinksinpathM[[i, j]], 2];

TelementsM = Table[Telements[i, j], {i, 1}, {j, 1}];
```

Service provider endpoints and VNF3

```
enodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathseM[[i, j, k]], 1], -1],
      {k, Length[CpathseM[[i, j]]]}]), {2}];

elinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathseM[[i, j, k]], -1], Drop[
          CpathseM[[i, j, k]], 1]}], 1], {k, Length[CpathseM[[i, j]]]}]), {2}];

enodesinpathsM = Table[enodesinpaths[i, j], {i, 1}, {j, 1}];

elinksinpathM = Table[elinksinpath[i, j], {i, 1}, {j, 1}];

Eelements[i_, j_] := Join[enodesinpathsM[[i, j]], elinksinpathM[[i, j]], 2];

EelementsM = Table[Eelements[i, j], {i, 1}, {j, 1}];
```

Endpoints connect to MANO

Service provider endpoint and MANO

```
dmanonodesinpaths[i_, j_] := Map[n# &,
    (Table[Drop[SpathsM[[i, j, k]], 1], {k, Length[SpathsM[[i, j]]]}]), {2}];
```

```
dmanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
   (Table[Drop[Transpose[{Drop[SpathsM[[i, j, k]], -1],
       Drop[SpathsM[[i, j, k]], 1]}], 1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanonodesinpathsM = Table[dmanonodesinpaths[i, j], {i, 1}, {j, 1}];

dmanolinksinpathM = Table[dmanolinksinpath[i, j], {i, 1}, {j, 1}];

Dmanoelements[i_, j_] :=
  Join[dmanonodesinpathsM[[i, j]], dmanolinksinpathM [[i, j]], 2];

DmanoelementsM = Table[Dmanoelements[i, j], {i, 1}, {j, 1}];
```

User endpoint and MANO

```
omanonodesinpaths[i_, j_] := Map[n# &,
   (Table[Drop[UpathsM[[i, j, k]], -1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
   (Table[Drop[Transpose[{Drop[UpathsM[[i, j, k]], -1],
       Drop[UpathsM[[i, j, k]], 1]}], -1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanonodesinpathsM = Table[omanonodesinpaths[i, j], {i, 1}, {j, 1}];

omanolinksinpathM = Table[omanolinksinpath[i, j], {i, 1}, {j, 1}];

Omanoelements[i_, j_] :=
  Join[omanonodesinpathsM[[i, j]], omanolinksinpathM [[i, j]], 2];

OmanoelementsM = Table[Omanoelements[i, j], {i, 1}, {j, 1}];
```

VNFS connect to MANO

VNF1 TO MANO

```
vnff[i_, j_] := Map[n# &,
   (Table[Drop[VpathsfM[[i, j, k]], 1], {k, Length[VpathsfM[[i, j]]]}]), {2}];

linksinf[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathsfM[[i, j, k]], -1], Drop[VpathsfM[[i, j, k]], 1]}],
     {k, Length[VpathsfM[[i, j]]]}]), {2}];

vnffM = Table[vnff[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksinfM = Table[linksinf[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementsf[i_, j_] := Join[vnffM[[i, j]], linksinfM[[i, j]], 2];

VelementsfM = Table[Velementsf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF2 TO MANO

```
vnfs[i_, j_] := Map[n# &,
   (Table[Drop[VpathssM[[i, j, k]], 1], {k, Length[VpathssM[[i, j]]]}]), {2}];
```

```
linksins[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathssM[[i, j, k]], -1], Drop[VpathssM[[i, j, k]], 1]}],
     {k, Length[VpathssM[[i, j]]]}]), {2}];

vnfsM = Table[vnfs[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksinsM = Table[linksins[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementss[i_, j_] := Join[vnfsM[[i, j]], linksinsM[[i, j]], 2];

VelementssM = Table[Velementss[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF3 TO MANO

```
vnft[i_, j_] := Map[n# &,
   (Table[Drop[VpathstM[[i, j, k]], 1], {k, Length[VpathstM[[i, j]]]}]), {2}];

linksint[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathstM[[i, j, k]], -1], Drop[VpathstM[[i, j, k]], 1]}],
     {k, Length[VpathstM[[i, j]]]}]), {2}];

vnftM = Table[vnft[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksintM = Table[linksint[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementst[i_, j_] := Join[vnftM[[i, j]], linksintM[[i, j]], 2];

VelementstM = Table[Velementst[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

The structure function and its analysis

convers the sets of elements providing a working path and control to a Boolean expression for reduction & analysis.

Service chain .

VNF1 and user endpoint.

```
Φf[i_, j_] := Table[{FelementsM[[i, j, k]] /. List → And},
   {k, Length[FelementsM[[i, j]]]}] /. List → Or

ΦfM = Table[Φf[i, j], {i, 1}, {j, 1}];

ΦfT = Table[{ΦfM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF2 and VNF1

```
Φs[i_, j_] := Table[{SelementsM[[i, j, k]] /. List → And},
   {k, Length[SelementsM[[i, j]]]}] /. List → Or

ΦsM = Table[Φs[i, j], {i, 1}, {j, 1}];

ΦsT = Table[{ΦsM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF3 and VNF2

```
Φt[i_, j_] := Table[{TelementsM[[i, j, k]] /. List → And},
   {k, Length[TelementsM[[i, j]]]}] /. List → Or
```

```
ΦtM = Table[Φt[i, j], {i, 1}, {j, 1}];
```

```
ΦtT = Table[{ΦtM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF3 and endpoint service provider

```
Φe[i_, j_] := Table[{EelementsM[[i, j, k]] /. List → And},
   {k, Length[EelementsM[[i, j]]]}] /. List → Or
```

```
ΦeM = Table[Φt[i, j], {i, 1}, {j, 1}];
```

```
ΦeT = Table[{ΦeM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

Endpoints connect to MANO

MANO and endpoint service provider

```
Φd[i_, j_] := Table[{DmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[DmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦdM = Table[Φd[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
ΦdT = Table[{ΦdM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

MANO and endpoint service provider

```
Φo[i_, j_] := Table[{OmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[OmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦoM = Table[Φo[i, j], {i, 1, NoOfMANO}, {j, 1, NoOfPoP}];
```

```
ΦoT = Table[{ΦoM[[k]] /. List → Or}, {k, NoOfMANO}] /. List → And;
```

MANO-> VNFs

VNFs-> MANO

VNF1 TO MANO

```
Φvf[i_, j_] := Table[{VelementsfM[[i, j, k]] /. List → And},
   {k, Length[VelementsfM[[i, j]]]}] /. List → Or
```

```
ΦvfM = Table[Φvf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvfT = Table[{ΦvfM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF2 TO MANO

```
Φvs[i_, j_] := Table[{VelementssM[[i, j, k]] /. List → And},
   {k, Length[VelementssM[[i, j]]]}] /. List → Or
```

```
ΦvsM = Table[Φvs[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvsT = Table[{ΦvsM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF3 TO MANO

```
Φvt[i_, j_] := Table[{VelementstM[[i, j, k]] /. List → And},
   {k, Length[VelementstM[[i, j]]]}] /. List → Or
```

```
ΦvtM = Table[Φvt[i, j], {i, 1}, {j, 1, NoOfMANO}];

ΦvtT = Table[{ΦvtM[[k]] /. List → Or}, {k, 1}] /. List → And;


Φnfv = ΦfT && ΦsT && ΦeT && ΦtT && ΦvfT && ΦdT && ΦoT && ΦvsT && ΦvtT // Simplify;
```

Extracting the minimum paths and cut

The minimum path and cut sets are extracted from the above logical expressions by replacing the heads of the expressions. The And and Or head are replaced by List heads. The not operation are replaced by identity and removed. Be aware that the operation may twist the meaning unless the logical expand produces results on standard form.

Functions

```
MinPaths[s_] := BooleanConvert[s] /. {And -> List, Or -> List}

MinCuts[s_] :=
 BooleanConvert[s, "CNF" ] /. {And -> List, Or -> List, Not -> Identity}
```

NFV

```
NFVminpath = MinPaths[Φnfv];

NFVmincut = MinCuts[Φnfv];

Export["D://NTNU 2 year//Master thesis//structure
    analysis//data//NFVmincut_norway_3e.mx", NFVmincut, "MX"]
```

```
D://NTNU 2 year//Master
  thesis//structure analysis//data//NFVmincut_norway_3e.mx
```

Cardinality cut sets.

```
Scenario3 = Length /@ (If[Head[#] =!= List, {#}, #] & /@ NFVmincut);

BinCounts[%, {1, Max[%] + 1, 1}]
```

```
{5, 10, 8, 120, 354, 643, 709, 525, 394, 183, 119, 32, 11, 4}
```

```
Position[Scenario3, _?(# < 2 &)]
```

```
{{844}, {845}, {846}, {2947}, {2949}}
```

```
Position[Scenario3, _?(1 < # < 3 &)]
```

```
{{1}, {348}, {765}, {2271}, {2547}, {2626}, {2705}, {2946}, {2948}, {3117}}
```

Appendix D

Structural analysis script of Scenario 3 in Mathematica

Structure model for Scenario 3

Define network

```
Primary = Graph[{TRD ⟷ OSL1, TRD ⟷ OSL2,
     TRD ⟷ BRG, TRD ⟷ STV, OSL1 ⟷ OSL2, OSL1 ⟷ STV, STV ⟷ BRG}];
Secondary = Graph[{TRDb ⟷ OSL2b, TRDb ⟷ BRGb, TRDb ⟷ STVb,
     OSL1b ⟷ OSL2b, OSL1b ⟷ STVb, STVb ⟷ BRGb}];
ConnectionPS = Graph[{TRD ⟷ TRDb, OSL1 ⟷ OSL1b,
     OSL2 ⟷ OSL2b, BRG ⟷ BRGb, STV ⟷ STVb}];
network = GraphUnion[Primary, Secondary, ConnectionPS, VertexLabels → "Name"];
```

A infrastructure node(a datacenter) with NFV MANO connects to BRG and BRGb nodes.

```
datacenter1 = Graph[{DC1 ⟷ TRD, DC1 ⟷ TRDb}, VertexLabels → "Name"];
datacenter2 = Graph[
    {OSL2 ⟷ DC2, OSL1 ⟷ DC2, OSL1b ⟷ DC2, OSL2b ⟷ DC2}, VertexLabels → "Name"];

datacenter3 =
  Graph[{DC3 ⟷ BRG, DC3 ⟷ BRGb, DC3 ⟷ STV, DC3 ⟷ STVb}, VertexLabels → "Name"];

vnf1 = Graph[{VNF1 ⟷ DC1}, VertexLabels → "Name"];

vnf2 = Graph[{VNF2 ⟷ DC1, VNF2 ⟷ DC2}, VertexLabels → "Name"];

vnf3 = Graph[{VNF3 ⟷ DC3, VNF3 ⟷ DC2}, VertexLabels → "Name"];

mano1 = Graph[{BRG → MANO1, BRGb → MANO1}, VertexLabels → "Name"];

Datacenter =
  GraphUnion[network, datacenter1, datacenter3, VertexShapeFunction → "Name"];

NFV = GraphUnion[network, datacenter1, datacenter3,
    vnf1, vnf2, vnf3, mano1, VertexShapeFunction → "Name"];

NFV1 =
  GraphUnion[network, datacenter1, vnf1, mano1, VertexShapeFunction → "Name"];

NFV2 = GraphUnion[network, datacenter1,
    datacenter2, vnf2, mano1, VertexShapeFunction → "Name"];

NFV3 = GraphUnion[network, datacenter2,
    datacenter3, vnf3, mano1, VertexShapeFunction → "Name"];
```

Connect to endpoints.

```
endpoint1 = Graph[{O1 → TRD, O1 → TRDb}, VertexLabels → "Name"];

endpoint2 = Graph[{D1 → STV, D1 → STVb}, VertexLabels → "Name"];

box1 = GraphUnion[network, endpoint2, datacenter1, vnf1, VertexLabels → "Name"];

box3 = GraphUnion[network, datacenter1,
    datacenter2, vnf2, vnf1, VertexLabels → "Name"];

box2 = GraphUnion[network, datacenter1,
    datacenter2, datacenter3, vnf2, vnf3, VertexLabels → "Name"];
```

```
box4 = GraphUnion[network, datacenter3,
    datacenter2, vnf3, endpoint1, VertexLabels → "Name"];
```

```
box5 = GraphUnion[network, endpoint1, mano1, endpoint2, VertexLabels → "Name"];
```

Network with endpoints only.

```
endpointsnetwork = GraphUnion[Datacenter, endpoint1, endpoint2];
```

```
HighlightGraph[endpointsnetwork,
  {network, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Network with endpoints and NFV elements.

```
overallNetwork = GraphUnion[NFV, endpoint1, endpoint2];
```

```
HighlightGraph[overallNetwork, {network, mano1, vnf1,
    vnf2, vnf3, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Find the path

Endpoints

```
Org[i_] := ToExpression["O" <> ToString[i]];
```

```
Dest[i_] := ToExpression["D" <> ToString[i]];
```

Datacenters

```
box[i_] := ToExpression["DC" <> ToString[i]];
```

VNFs

```
Vir[i_] := ToExpression["VNF" <> ToString[i]];
```

MANO

```
Ctrl[i_] := ToExpression["MANO" <> ToString[i]];
```

Paths between endpoints(service provider) to endpoints(user) include the service function chain.

```
Cpathsf[i_, j_] := FindPath[box1, Dest[j], Vir[1], Infinity, All];
```

```
Cpathss[i_, j_] := FindPath[box3, Vir[2], Vir[1], Infinity, All];
```

```
Cpathst[i_, j_] := FindPath[box2, Vir[3], Vir[2], Infinity, All];
```

```
Cpathse[i_, j_] := FindPath[box4, Org[i], Vir[3], Infinity, All];
```

Paths between endpoints and MANO

```
Spaths[i_, j_] := FindPath[box5, Org[i], Ctrl[j], Infinity, All];
```

```
Upaths[i_, j_] := FindPath[box5, Dest[i], Ctrl[j], Infinity, All];
```

Paths between VNFs and MANO

```
Vpathsf[i_, j_] := FindPath[NFV1, Vir[1], Ctrl[j], Infinity, All];
```

```
Vpathss[i_, j_] := FindPath[NFV2, Vir[2], Ctrl[j], Infinity, All];
```

```
Vpathst[i_, j_] := FindPath[NFV3, Vir[3], Ctrl[j], Infinity, All];
```

Define parameters.

```
NoOfPoP = 1;  NoOfVNF = 3; NoOfMANO = 1;
```

```
CpathsfM = Table[Cpathsf[i, j], {i, 1}, {j, 1, NoOfPoP}];
```

```
CpathssM = Table[Cpathss[i, j], {i, 1}, {j, 1}];
```

```
CpathstM = Table[Cpathst[i, j], {i, 1}, {j, 1}];
```

```
CpathseM = Table[Cpathse[i, j], {i, 1}, {j, 1}];
```

```
SpathsM = Table[Spaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
UpathsM = Table[Upaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
VpathsfM = Table[Vpathsf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
VpathssM = Table[Vpathss[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
VpathstM = Table[Vpathst[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

Converting paths into network elements

```
VMap = {TRD, TRDb, OSL1, OSL1b, OSL2, OSL2b,
   STV, STVb, BRG, BRGb, DC1, DC3, DC2, MANO1, VNF1, VNF2, VNF3};
```

Service chain .

User endpoint and VNF1.

```
fnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathsfM[[i, j, k]], 1], -1],
      {k, Length[CpathsfM[[i, j]]]}]), {2}];
```

```
flinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathsfM[[i, j, k]], -1], Drop[
          CpathsfM[[i, j, k]], 1]}], 1], {k, Length[CpathsfM[[i, j]]]}]), {2}];
```

```
fnodesinpathsM = Table[fnodesinpaths[i, j], {i, 1}, {j, 1}];
```

```
flinksinpathM = Table[flinksinpath[i, j], {i, 1}, {j, 1}];
```

```
Felements[i_, j_] := Join[fnodesinpathsM[[i, j]], flinksinpathM[[i, j]], 2];
```

```
FelementsM = Table[Felements[i, j], {i, 1}, {j, 1}];
```

VNF2 and VNF1

```
snodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathssM[[i, j, k]], 1], -1],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
slinksinpath[i_, j_] := Map[If[Position[VMap, First[#]][[1, 1]] <
       Position[VMap, Last[#]][[1, 1]], Subscript[l, First[#], Last[#]],
      Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathssM[[i, j, k]], -1], Drop[CpathssM[[i, j, k]], 1]}],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
snodesinpathsM = Table[snodesinpaths[i, j], {i, 1}, {j, 1}];
```

```
slinksinpathM = Table[slinksinpath[i, j], {i, 1}, {j, 1}];

Selements[i_, j_] := Join[snodesinpathsM[[i, j]], slinksinpathM[[i, j]], 2];
SelementsM = Table[Selements[i, j], {i, 1}, {j, 1}];
```

VNF3 and VNF2

```
tnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathstM[[i, j, k]], 1], -1],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tlinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathstM[[i, j, k]], -1], Drop[CpathstM[[i, j, k]], 1]}],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tnodesinpathsM = Table[tnodesinpaths[i, j], {i, 1}, {j, 1}];

tlinksinpathM = Table[tlinksinpath[i, j], {i, 1}, {j, 1}];

Telements[i_, j_] := Join[tnodesinpathsM[[i, j]], tlinksinpathM[[i, j]], 2];

TelementsM = Table[Telements[i, j], {i, 1}, {j, 1}];
```

Service provider endpoints and VNF3

```
enodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathseM[[i, j, k]], 1], -1],
      {k, Length[CpathseM[[i, j]]]}]), {2}];

elinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathseM[[i, j, k]], -1], Drop[
          CpathseM[[i, j, k]], 1]}], 1], {k, Length[CpathseM[[i, j]]]}]), {2}];

enodesinpathsM = Table[enodesinpaths[i, j], {i, 1}, {j, 1}];

elinksinpathM = Table[elinksinpath[i, j], {i, 1}, {j, 1}];

Eelements[i_, j_] := Join[enodesinpathsM[[i, j]], elinksinpathM[[i, j]], 2];

EelementsM = Table[Eelements[i, j], {i, 1}, {j, 1}];
```

Endpoints connect to MANO

Service provider endpoint and MANO

```
dmanonodesinpaths[i_, j_] := Map[n# &,
    (Table[Drop[SpathsM[[i, j, k]], 1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      l_{First[#],Last[#]}, l_{Last[#],First[#]}] &, (Table[
      Drop[Transpose[{Drop[SpathsM[[i, j, k]], -1], Drop[SpathsM[[i, j, k]], 1]}],
       1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanonodesinpathsM =
  Table[dmanonodesinpaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];

dmanolinksinpathM =
  Table[dmanolinksinpath[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
Dmanoelements[i_, j_] :=
  Join[dmanonodesinpathsM[[i, j]], dmanolinksinpathM [[i, j]], 2];

DmanoelementsM = Table[Dmanoelements[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

User endpoint and MANO

```
omanonodesinpaths[i_, j_] := Map[n# &,
   (Table[Drop[UpathsM[[i, j, k]], 1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
   (Table[Drop[Transpose[{Drop[UpathsM[[i, j, k]], -1],
        Drop[UpathsM[[i, j, k]], 1]}], 1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanonodesinpathsM = Table[omanonodesinpaths[i, j], {i, 1}, {j, 1, NoOfMANO}];

omanolinksinpathM = Table[omanolinksinpath[i, j], {i, 1}, {j, 1, NoOfMANO}];

Omanoelements[i_, j_] :=
  Join[omanonodesinpathsM[[i, j]], omanolinksinpathM [[i, j]], 2];

OmanoelementsM = Table[Omanoelements[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNFS connect to MANO

VNF1 TO MANO

```
vnff[i_, j_] := Map[n# &,
   (Table[Drop[VpathsfM[[i, j, k]], 1], {k, Length[VpathsfM[[i, j]]]}]), {2}];

linksinf[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathsfM[[i, j, k]], -1], Drop[VpathsfM[[i, j, k]], 1]}],
     {k, Length[VpathsfM[[i, j]]]}]), {2}];

vnffM = Table[vnff[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksinfM = Table[linksinf[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementsf[i_, j_] := Join[vnffM[[i, j]], linksinfM[[i, j]], 2];

VelementsfM = Table[Velementsf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF2 TO MANO

```
vnfs[i_, j_] := Map[n# &,
   (Table[Drop[VpathssM[[i, j, k]], 1], {k, Length[VpathssM[[i, j]]]}]), {2}];

linksins[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathssM[[i, j, k]], -1], Drop[VpathssM[[i, j, k]], 1]}],
     {k, Length[VpathssM[[i, j]]]}]), {2}];

vnfsM = Table[vnfs[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksinsM = Table[linksins[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
Velementss[i_, j_] := Join[vnfsM[[i, j]], linksinsM[[i, j]], 2];

VelementssM = Table[Velementss[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF3 TO MANO

```
vnft[i_, j_] := Map[n# &,
    (Table[Drop[VpathstM[[i, j, k]], 1], {k, Length[VpathstM[[i, j]]]}]), {2}];

linksint[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[VpathstM[[i, j, k]], -1], Drop[VpathstM[[i, j, k]], 1]}],
      {k, Length[VpathstM[[i, j]]]}]), {2}];

vnftM = Table[vnft[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksintM = Table[linksint[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementst[i_, j_] := Join[vnftM[[i, j]], linksintM[[i, j]], 2];

VelementstM = Table[Velementst[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

The structure function and its analysis

convers the sets of elements providing a working path and control to a Boolean expression for reduction & analysis.

Service chain .

VNF1 and user endpoint.

```
Φf[i_, j_] := Table[{FelementsM[[i, j, k]] /. List → And},
    {k, Length[FelementsM[[i, j]]]}] /. List → Or

ΦfM = Table[Φf[i, j], {i, 1}, {j, 1}];

ΦfT = Table[{ΦfM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF2 and VNF1

```
Φs[i_, j_] := Table[{SelementsM[[i, j, k]] /. List → And},
    {k, Length[SelementsM[[i, j]]]}] /. List → Or

ΦsM = Table[Φs[i, j], {i, 1}, {j, 1}];

ΦsT = Table[{ΦsM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF3 and VNF2

```
Φt[i_, j_] := Table[{TelementsM[[i, j, k]] /. List → And},
    {k, Length[TelementsM[[i, j]]]}] /. List → Or

ΦtM = Table[Φt[i, j], {i, 1}, {j, 1}];

ΦtT = Table[{ΦtM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF3 and endpoint service provider

```
Φe[i_, j_] := Table[{EelementsM[[i, j, k]] /. List → And},
   {k, Length[EelementsM[[i, j]]]}] /. List → Or
```

```
ΦeM = Table[Φt[i, j], {i, 1}, {j, 1}];
```

```
ΦeT = Table[{ΦeM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

Endpoints connect to MANO

MANO and endpoint service provider

```
Φd[i_, j_] := Table[{DmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[DmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦdM = Table[Φd[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
ΦdT = Table[{ΦdM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

MANO and endpoint service provider

```
Φo[i_, j_] := Table[{OmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[OmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦoM = Table[Φo[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
ΦoT = Table[{ΦoM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNFs-> MANO

VNF1 TO MANO

```
Φvf[i_, j_] := Table[{VelementsfM[[i, j, k]] /. List → And},
   {k, Length[VelementsfM[[i, j]]]}] /. List → Or
```

```
ΦvfM = Table[Φvf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvfT = Table[{ΦvfM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF2 TO MANO

```
Φvs[i_, j_] := Table[{VelementssM[[i, j, k]] /. List → And},
   {k, Length[VelementssM[[i, j]]]}] /. List → Or
```

```
ΦvsM = Table[Φvs[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvsT = Table[{ΦvsM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF3 TO MANO

```
Φvt[i_, j_] := Table[{VelementstM[[i, j, k]] /. List → And},
   {k, Length[VelementstM[[i, j]]]}] /. List → Or
```

```
ΦvtM = Table[Φvt[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvtT = Table[{ΦvtM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

```
Φnfv = ΦfT && ΦsT && ΦeT && ΦtT && ΦvfT && ΦdT && ΦoT && ΦvsT && ΦvtT // Simplify;
```

Extracting the minimum paths and cut

The minimum path and cut sets are extracted from the above logical expressions by replacing the heads of the expressions. The And and Or head are replaced by List heads. The not operation are replaced by identity and removed. Be aware that the operation may twist the meaning unless the logical expand produces results on standard form.

Functions

```
MinPaths[s_] := BooleanConvert[s] /. {And -> List, Or -> List}
```

```
MinCuts[s_] :=
 BooleanConvert[s, "CNF"] /. {And -> List, Or -> List, Not -> Identity}
```

NFV

```
NFVminpath = MinPaths[Φnfv];
```

```
NFVmincut = MinCuts[Φnfv];
```

```
Export["D://NTNU 2 year//Master thesis//structure
   analysis//data//NFVmincut_norway_4e.mx", NFVmincut, "MX"]
```

```
D://NTNU 2 year//Master
  thesis//structure analysis//data//NFVmincut_norway_4e.mx
```

Cardinality cut sets.

```
Scenario4 = Length /@ (If[Head[#] =!= List, {#}, #] & /@ NFVmincut);
```

```
BinCounts[%, {1, Max[%] + 1, 1}]
```

```
{3, 15, 8, 65, 431, 766, 1038, 1389, 1431,
 1619, 1559, 1198, 875, 532, 268, 142, 60, 13, 3}
```

```
Position[Scenario4, _?(# < 2 &)]
```

```
{{2774}, {4289}, {11 164}}
```

```
Position[Scenario4, _?(1 < # < 3 &)]
```

```
{{1}, {1303}, {2669}, {2775}, {3321}, {3322}, {4213}, {9132},
 {9867}, {10 095}, {10 318}, {11 163}, {11 165}, {11 241}, {11 415}}
```

Appendix E

Structural analysis script of Scenario 4 in Mathematica

Structure model of Scenario 4

Define network

```
Primary = Graph[{TRD ⟷ OSL1, TRD ⟷ OSL2,
     TRD ⟷ BRG, TRD ⟷ STV, OSL1 ⟷ OSL2, OSL1 ⟷ STV, STV ⟷ BRG}];
Secondary = Graph[{TRDb ⟷ OSL2b, TRDb ⟷ BRGb, TRDb ⟷ STVb,
     OSL1b ⟷ OSL2b, OSL1b ⟷ STVb, STVb ⟷ BRGb}];
ConnectionPS = Graph[{TRD ⟷ TRDb, OSL1 ⟷ OSL1b,
     OSL2 ⟷ OSL2b, BRG ⟷ BRGb, STV ⟷ STVb}];
network = GraphUnion[Primary, Secondary, ConnectionPS, VertexLabels → "Name"];

datacenter1 = Graph[{DC1 ⟷ TRD, DC1 ⟷ TRDb}, VertexLabels → "Name"];

mano1 = Graph[{BRG → MANO1, BRGb → MANO1}, VertexLabels → "Name"];
mano2 = Graph[{TRD → MANO2, TRDb → MANO2}, VertexLabels → "Name"];


datacenter2 = Graph[
    {DC2 ⟷ OSL1, DC2 ⟷ OSL1b, DC2 ⟷ OSL2b, DC2 ⟷ OSL2}, VertexLabels → "Name"];

vnf1 = Graph[{VNF1 ⟷ DC1}, VertexLabels → "Name"];

vnf2 = Graph[{VNF2 ⟷ DC1}, VertexLabels → "Name"];

vnf3 = Graph[{VNF3 ⟷ DC2}, VertexLabels → "Name"];




Datacenter =
  GraphUnion[network, datacenter1, datacenter2, VertexShapeFunction → "Name"];

NFV = GraphUnion[network, datacenter1, datacenter2,
    vnf1, vnf2, vnf3, mano1, mano2, VertexShapeFunction → "Name"];

NFV2 = GraphUnion[network, datacenter1,
    mano1, vnf1, vnf2, mano2, VertexShapeFunction → "Name"];

NFV3 = GraphUnion[network, mano1, vnf3,
    datacenter2, mano2, VertexShapeFunction → "Name"];
```

Connect to endpoints.

```
endpoint1 = Graph[{O1 → TRD, O1 → TRDb}, VertexLabels → "Name"];

endpoint2 = Graph[{D1 → STV, D1 → STVb}, VertexLabels → "Name"];
```

Network with endpoints only.

```
NFV1 = GraphUnion[network, mano2, mano1,
    endpoint1, endpoint2, VertexShapeFunction → "Name"];

endpointsnetwork1 = GraphUnion[datacenter1, endpoint2, vnf1, network];

endpointsnetwork2 = GraphUnion[datacenter1, datacenter2, vnf1, vnf2, network];

endpointsnetwork3 = GraphUnion[datacenter2, endpoint1, vnf3, network];
```

```
endpointsnetwork = GraphUnion[network, endpoint1, endpoint2];

HighlightGraph[endpointsnetwork,
  {network, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Network with endpoints and NFV elements.

```
overallNetwork = GraphUnion[NFV, endpoint1, endpoint2];

HighlightGraph[overallNetwork, {network, mano1, mano2,
   vnf1, vnf2, vnf3, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Find the path

Endpoints

```
Org[i_] := ToExpression["O" <> ToString[i]];

Dest[i_] := ToExpression["D" <> ToString[i]];
```

Datacenters

```
box[i_] := ToExpression["DC" <> ToString[i]];
```

VNFs

```
Vir[i_] := ToExpression["VNF" <> ToString[i]];
```

MANO

```
Ctrl[i_] := ToExpression["MANO" <> ToString[i]];
```

Paths between endpoints(service provider) to endpoints(user) include the service function chain.

```
Cpathsf[i_, j_] := FindPath[endpointsnetwork1, Dest[1], Vir[1], Infinity, All]

Cpathss[i_, j_] := FindPath[endpointsnetwork2, Vir[2], Vir[1], Infinity, All]

Cpathst[i_, j_] := FindPath[overallNetwork, Vir[3], Vir[2], Infinity, All]

Cpathse[i_, j_] := FindPath[endpointsnetwork3, Org[i], Vir[3], Infinity, All]
```

Paths between endpoints and MANO

```
Spaths[i_, j_] := FindPath[NFV1, Org[i], Ctrl[j], Infinity, All];

Upaths[i_, j_] := FindPath[NFV1, Dest[i], Ctrl[j], Infinity, All];
```

Paths between VNFs and MANO

```
Vpathsf[i_, j_] := FindPath[NFV2, Vir[1], Ctrl[j], Infinity, All];

Vpathss[i_, j_] := FindPath[NFV2, Vir[2], Ctrl[j], Infinity, All];

Vpathst[i_, j_] := FindPath[NFV3, Vir[3], Ctrl[j], Infinity, All];
```

Define parameters.

```
NoOfPoP = 1;  NoOfVNF = 3; NoOfMANO = 2;
```

```
CpathsfM = Table[Cpathsf[i, j], {i, 1}, {j, 1}];

CpathssM = Table[Cpathss[i, j], {i, 1}, {j, 1}];

CpathstM = Table[Cpathst[i, j], {i, 1}, {j, 1}];

CpathseM = Table[Cpathse[i, j], {i, 1}, {j, 1}];

SpathsM = Table[Spaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];

UpathsM = Table[Upaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];

VpathsfM = Table[Vpathsf[i, j], {i, 1}, {j, 1, NoOfMANO}];

VpathssM = Table[Vpathss[i, j], {i, 1}, {j, 1, NoOfMANO}];

VpathstM = Table[Vpathst[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

Converting paths into network elements

```
VMap = {TRD, TRDb, OSL1, OSL1b, OSL2, OSL2b, STV,
    STVb, BRG, BRGb, DC1, DC2, MANO1, MANO2, VNF1, VNF2, VNF3};
```

Service chain .

User endpoint and VNF1.

```
fnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathsfM[[i, j, k]], 1], -1],
      {k, Length[CpathsfM[[i, j]]]}]), {2}];

flinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathsfM[[i, j, k]], -1], Drop[
          CpathsfM[[i, j, k]], 1]}], 1], {k, Length[CpathsfM[[i, j]]]}]), {2}];

fnodesinpathsM = Table[fnodesinpaths[i, j], {i, 1}, {j, 1}];

flinksinpathM = Table[flinksinpath[i, j], {i, 1}, {j, 1}];

Felements[i_, j_] := Join[fnodesinpathsM[[i, j]], flinksinpathM[[i, j]], 2];

FelementsM = Table[Felements[i, j], {i, 1}, {j, 1}];
```

VNF2 and VNF1

```
snodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathssM[[i, j, k]], 1], -1],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
slinksinpath[i_, j_] := Map[If[Position[VMap, First[#]][[1, 1]] <
       Position[VMap, Last[#]][[1, 1]], Subscript[l, First[#], Last[#]],
      Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathssM[[i, j, k]], -1], Drop[CpathssM[[i, j, k]], 1]}],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
snodesinpathsM = Table[snodesinpaths[i, j], {i, 1}, {j, 1}];

slinksinpathM = Table[slinksinpath[i, j], {i, 1}, {j, 1}];
```

```
Selements[i_, j_] := Join[snodesinpathsM[[i, j]], slinksinpathM[[i, j]], 2];
SelementsM = Table[Selements[i, j], {i, 1}, {j, 1}];
```

VNF3 and VNF2

```
tnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathstM[[i, j, k]], 1], -1],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tlinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathstM[[i, j, k]], -1], Drop[CpathstM[[i, j, k]], 1]}],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tnodesinpathsM = Table[tnodesinpaths[i, j], {i, 1}, {j, 1}];

tlinksinpathM = Table[tlinksinpath[i, j], {i, 1}, {j, 1}];

Telements[i_, j_] := Join[tnodesinpathsM[[i, j]], tlinksinpathM[[i, j]], 2];

TelementsM = Table[Telements[i, j], {i, 1}, {j, 1}];
```

Service provider endpoints and VNF3

```
enodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathseM[[i, j, k]], 1], -1],
      {k, Length[CpathseM[[i, j]]]}]), {2}];

elinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathseM[[i, j, k]], -1], Drop[
          CpathseM[[i, j, k]], 1]}], 1], {k, Length[CpathseM[[i, j]]]}]), {2}];

enodesinpathsM = Table[enodesinpaths[i, j], {i, 1}, {j, 1}];

elinksinpathM = Table[elinksinpath[i, j], {i, 1}, {j, 1}];

Eelements[i_, j_] := Join[enodesinpathsM[[i, j]], elinksinpathM[[i, j]], 2];

EelementsM = Table[Eelements[i, j], {i, 1}, {j, 1}];
```

Endpoints connect to MANO

Service provider endpoint and MANO

```
dmanonodesinpaths[i_, j_] := Map[n# &,
    (Table[Drop[SpathsM[[i, j, k]], 1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      l_{First[#],Last[#]}, l_{Last[#],First[#]}] &, (Table[
      Drop[Transpose[{Drop[SpathsM[[i, j, k]], -1], Drop[SpathsM[[i, j, k]], 1]}],
       1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanonodesinpathsM =
  Table[dmanonodesinpaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];

dmanolinksinpathM =
  Table[dmanolinksinpath[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
Dmanoelements[i_, j_] :=
  Join[dmanonodesinpathsM[[i, j]], dmanolinksinpathM [[i, j]], 2];

DmanoelementsM = Table[Dmanoelements[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

User endpoint and MANO

```
omanonodesinpaths[i_, j_] := Map[n# &,
   (Table[Drop[UpathsM[[i, j, k]], 1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
   (Table[Drop[Transpose[{Drop[UpathsM[[i, j, k]], -1],
        Drop[UpathsM[[i, j, k]], 1]}], 1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanonodesinpathsM = Table[omanonodesinpaths[i, j], {i, 1}, {j, 1, NoOfMANO}];

omanolinksinpathM = Table[omanolinksinpath[i, j], {i, 1}, {j, 1, NoOfMANO}];

Omanoelements[i_, j_] :=
  Join[omanonodesinpathsM[[i, j]], omanolinksinpathM [[i, j]], 2];

OmanoelementsM = Table[Omanoelements[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNFS connect to MANO

VNF1 TO MANO

```
vnff[i_, j_] := Map[n# &,
   (Table[Drop[VpathsfM[[i, j, k]], 1], {k, Length[VpathsfM[[i, j]]]}]), {2}];

linksinf[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathsfM[[i, j, k]], -1], Drop[VpathsfM[[i, j, k]], 1]}],
     {k, Length[VpathsfM[[i, j]]]}]), {2}];

vnffM = Table[vnff[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksinfM = Table[linksinf[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementsf[i_, j_] := Join[vnffM[[i, j]], linksinfM[[i, j]], 2];

VelementsfM = Table[Velementsf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF2 TO MANO

```
vnfs[i_, j_] := Map[n# &,
   (Table[Drop[VpathssM[[i, j, k]], 1], {k, Length[VpathssM[[i, j]]]}]), {2}];

linksins[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathssM[[i, j, k]], -1], Drop[VpathssM[[i, j, k]], 1]}],
     {k, Length[VpathssM[[i, j]]]}]), {2}];

vnfsM = Table[vnfs[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksinsM = Table[linksins[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
Velementss[i_, j_] := Join[vnfsM[[i, j]], linksinsM[[i, j]], 2];

VelementssM = Table[Velementss[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF3 TO MANO

```
vnft[i_, j_] := Map[n# &,
    (Table[Drop[VpathstM[[i, j, k]], 1], {k, Length[VpathstM[[i, j]]]}]), {2}];

linksint[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[VpathstM[[i, j, k]], -1], Drop[VpathstM[[i, j, k]], 1]}],
      {k, Length[VpathstM[[i, j]]]}]), {2}];

vnftM = Table[vnft[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksintM = Table[linksint[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementst[i_, j_] := Join[vnftM[[i, j]], linksintM[[i, j]], 2];

VelementstM = Table[Velementst[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

The structure function and its analysis

convers the sets of elements providing a working path and control to a Boolean expression for reduction & analysis.

Service chain .

VNF1 and user endpoint.

```
Φf[i_, j_] := Table[{FelementsM[[i, j, k]] /. List → And},
    {k, Length[FelementsM[[i, j]]]}] /. List → Or

ΦfM = Table[Φf[i, j], {i, 1}, {j, 1}];

ΦfT = Table[{ΦfM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF2 and VNF1

```
Φs[i_, j_] := Table[{SelementsM[[i, j, k]] /. List → And},
    {k, Length[SelementsM[[i, j]]]}] /. List → Or

ΦsM = Table[Φs[i, j], {i, 1}, {j, 1}];

ΦsT = Table[{ΦsM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF3 and VNF2

```
Φt[i_, j_] := Table[{TelementsM[[i, j, k]] /. List → And},
    {k, Length[TelementsM[[i, j]]]}] /. List → Or

ΦtM = Table[Φt[i, j], {i, 1}, {j, 1}];

ΦtT = Table[{ΦtM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF3 and endpoint service provider

```
Φe[i_, j_] := Table[{EelementsM[[i, j, k]] /. List → And},
   {k, Length[EelementsM[[i, j]]]}] /. List → Or
```

```
ΦeM = Table[Φt[i, j], {i, 1}, {j, 1}];
```

```
ΦeT = Table[{ΦeM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

Endpoints connect to MANO

MANO and endpoint service provider

```
Φd[i_, j_] := Table[{DmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[DmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦdM = Table[Φd[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
ΦdT = Table[{ΦdM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

MANO and endpoint service provider

```
Φo[i_, j_] := Table[{OmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[OmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦoM = Table[Φo[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
ΦoT = Table[{ΦoM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNFs-> MANO

VNF1 TO MANO

```
Φvf[i_, j_] := Table[{VelementsfM[[i, j, k]] /. List → And},
   {k, Length[VelementsfM[[i, j]]]}] /. List → Or
```

```
ΦvfM = Table[Φvf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvfT = Table[{ΦvfM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF2 TO MANO

```
Φvs[i_, j_] := Table[{VelementssM[[i, j, k]] /. List → And},
   {k, Length[VelementssM[[i, j]]]}] /. List → Or
```

```
ΦvsM = Table[Φvs[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvsT = Table[{ΦvsM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF3 TO MANO

```
Φvt[i_, j_] := Table[{VelementstM[[i, j, k]] /. List → And},
   {k, Length[VelementstM[[i, j]]]}] /. List → Or
```

```
ΦvtM = Table[Φvt[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvtT = Table[{ΦvtM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

```
Φnfv = ΦfT && ΦsT && ΦeT && ΦtT && ΦvfT && ΦdT && ΦoT && ΦvsT && ΦvtT // Simplify;
```

Extracting the minimum paths and cut

The minimum path and cut sets are extracted from the above logical expressions by replacing the heads of the expressions. The And and Or head are replaced by List heads. The not operation are replaced by identity and removed. Be aware that the operation may twist the meaning unless the logical expand produces results on standard form.

Functions

```
MinPaths[s_] := BooleanConvert[s] /. {And -> List, Or -> List}
```

```
MinCuts[s_] :=
 BooleanConvert[s, "CNF"] /. {And -> List, Or -> List, Not -> Identity}
```

NFV

```
NFVminpath = MinPaths[Φnfv];
```

```
NFVmincut = MinCuts[Φnfv];
```

```
Export["D://NTNU 2 year//Master thesis//structure
   analysis//data//NFVmincut_norway_5e.mx", NFVmincut, "MX"]
```

```
D://NTNU 2 year//Master
  thesis//structure analysis//data//NFVmincut_norway_5e.mx
```

Cardinality cut sets.

```
Scenario5 = Length /@ (If[Head[#] =!= List, {#}, #] & /@ NFVmincut);
```

```
BinCounts[%, {1, Max[%] + 1, 1}]
```

```
{5, 6, 7, 121, 335, 668, 1251, 1339, 1157, 752, 451, 236, 80, 51, 10, 5}
```

```
Position[Scenario5, _?(# < 2 &)]
```

```
{{2368}, {2369}, {6190}, {6191}, {6192}}
```

```
Position[Scenario5, _?(1 < # < 3 &)]
```

```
{{2370}, {5386}, {5760}, {5836}, {5912}, {6474}}
```

Appendix F

Structural analysis script of Scenario 5 in Mathematica

Structure model of Scenario 5

Define network

```
Primary = Graph[{TRD ⟷ OSL1, TRD ⟷ OSL2,
     TRD ⟷ BRG, TRD ⟷ STV, OSL1 ⟷ OSL2, OSL1 ⟷ STV, STV ⟷ BRG}];
Secondary = Graph[{TRDb ⟷ OSL2b, TRDb ⟷ BRGb, TRDb ⟷ STVb,
     OSL1b ⟷ OSL2b, OSL1b ⟷ STVb, STVb ⟷ BRGb}];
ConnectionPS = Graph[{TRD ⟷ TRDb, OSL1 ⟷ OSL1b,
     OSL2 ⟷ OSL2b, BRG ⟷ BRGb, STV ⟷ STVb}];
network = GraphUnion[Primary, Secondary, ConnectionPS, VertexLabels → "Name"];

datacenter1 = Graph[{DC1 ⟷ TRD, DC1 ⟷ TRDb}, VertexLabels → "Name"];
datacenter2 = Graph[
    {OSL2 ⟷ DC2, OSL1 ⟷ DC2, OSL1b ⟷ DC2, OSL2b ⟷ DC2}, VertexLabels → "Name"];

vnf1 = Graph[{VNF1 ⟷ DC1}, VertexLabels → "Name"];

vnf2 = Graph[{VNF2 ⟷ DC1, VNF2 ⟷ DC2}, VertexLabels → "Name"];

vnf3 = Graph[{VNF3 ⟷ DC2}, VertexLabels → "Name"];

mano2 = Graph[{TRD → MANO2, TRDb → MANO2}, VertexLabels → "Name"];

mano1 = Graph[{BRG → MANO1, BRGb → MANO1}, VertexLabels → "Name"];

Datacenter =
  GraphUnion[network, datacenter1, datacenter2, VertexShapeFunction → "Name"];

NFV = GraphUnion[network, datacenter1, datacenter2,
    vnf1, vnf2, vnf3, mano1, mano2, VertexShapeFunction → "Name"];

NFV1 = GraphUnion[network, datacenter1,
    vnf1, mano1, mano2, VertexShapeFunction → "Name"];

NFV2 = GraphUnion[network, datacenter1,
    datacenter2, vnf2, mano1, mano2, VertexShapeFunction → "Name"];

NFV3 = GraphUnion[network, datacenter1,
    datacenter2, vnf3, mano1, mano2, VertexShapeFunction → "Name"];
```

Connect to endpoints.

```
endpoint1 = Graph[{O1 → TRD, O1 → TRDb}, VertexLabels → "Name"];

endpoint2 = Graph[{D1 → STV, D1 → STVb}, VertexLabels → "Name"];

box1 = GraphUnion[network, datacenter1, endpoint2, vnf1, VertexLabels → "Name"];

box3 = GraphUnion[network, datacenter1, datacenter2,
    endpoint2, vnf3, vnf2, vnf1, VertexLabels → "Name"];

box4 = GraphUnion[network, datacenter2, endpoint1, vnf3, VertexLabels → "Name"];

box5 = GraphUnion[network, endpoint1, mano1,
    endpoint2, mano2, endpoint2, VertexLabels → "Name"];
```

Network with endpoints only.

```
endpointsnetwork = GraphUnion[Datacenter, endpoint1, endpoint2];
```

```
HighlightGraph[endpointsnetwork,
  {network, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Network with endpoints and NFV elements.

```
overallNetwork = GraphUnion[NFV, endpoint1, endpoint2];
```

```
HighlightGraph[overallNetwork, {network, mano1, mano2,
   vnf1, vnf2, vnf3, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Find the path

Endpoints

```
Org[i_] := ToExpression["O" <> ToString[i]];
```

```
Dest[i_] := ToExpression["D" <> ToString[i]];
```

Datacenters

```
box[i_] := ToExpression["DC" <> ToString[i]];
```

VNFs

```
Vir[i_] := ToExpression["VNF" <> ToString[i]];
```

MANO

```
Ctrl[i_] := ToExpression["MANO" <> ToString[i]];
```

Paths between endpoints(service provider) to endpoints(user) include the service function chain.

```
Cpathsf[i_, j_] := FindPath[box1, Dest[j], Vir[1], Infinity, All]
```

```
Cpathss[i_, j_] := FindPath[box3, Vir[2], Vir[1], Infinity, All]
```

```
Cpathst[i_, j_] := FindPath[box3, Vir[3], Vir[2], Infinity, All]
```

```
Cpathse[i_, j_] := FindPath[box4, Org[i], Vir[3], Infinity, All]
```

Paths between endpoints and MANO

```
Spaths[i_, j_] := FindPath[box5, Org[i], Ctrl[j], Infinity, All];
```

```
Upaths[i_, j_] := FindPath[box5, Dest[i], Ctrl[j], Infinity, All];
```

Paths between VNFs and MANO

```
Vpathsf[i_, j_] := FindPath[NFV1, Vir[1], Ctrl[j], Infinity, All];
```

```
Vpathss[i_, j_] := FindPath[NFV2, Vir[2], Ctrl[j], Infinity, All];
```

```
Vpathst[i_, j_] := FindPath[NFV3, Vir[3], Ctrl[j], Infinity, All];
```

Define parameters.

```
NoOfPoP = 1;  NoOfVNF = 3; NoOfMANO = 2;
```

```
CpathsfM = Table[Cpathsf[i, j], {i, 1}, {j, 1, NoOfPoP}];
```

```
CpathssM = Table[Cpathss[i, j], {i, 1}, {j, 1}];
```

```
CpathstM = Table[Cpathst[i, j], {i, 1}, {j, 1}];

CpathseM = Table[Cpathse[i, j], {i, 1}, {j, 1}];

SpathsM = Table[Spaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];

UpathsM = Table[Upaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];



VpathsfM = Table[Vpathsf[i, j], {i, 1}, {j, 1, NoOfMANO}];

VpathssM = Table[Vpathss[i, j], {i, 1}, {j, 1, NoOfMANO}];

VpathstM = Table[Vpathst[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

Converting paths into network elements

```
VMap = {TRD, TRDb, OSL1, OSL1b, OSL2, OSL2b, STV,
   STVb, BRG, BRGb, DC1, DC2, MANO1, MANO2, VNF1, VNF2, VNF3};
```

Service chain .

User endpoint and VNF1.

```
fnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathsfM[[i, j, k]], 1], -1],
      {k, Length[CpathsfM[[i, j]]]}]), {2}];

flinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathsfM[[i, j, k]], -1], Drop[
          CpathsfM[[i, j, k]], 1]}], 1], {k, Length[CpathsfM[[i, j]]]}]), {2}];

fnodesinpathsM = Table[fnodesinpaths[i, j], {i, 1}, {j, 1}];

flinksinpathM = Table[flinksinpath[i, j], {i, 1}, {j, 1}];

Felements[i_, j_] := Join[fnodesinpathsM[[i, j]], flinksinpathM[[i, j]], 2];

FelementsM = Table[Felements[i, j], {i, 1}, {j, 1}];
```

VNF2 and VNF1

```
snodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathssM[[i, j, k]], 1], -1],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
slinksinpath[i_, j_] := Map[If[Position[VMap, First[#]][[1, 1]] <
       Position[VMap, Last[#]][[1, 1]], Subscript[l, First[#], Last[#]],
      Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathssM[[i, j, k]], -1], Drop[CpathssM[[i, j, k]], 1]}],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
snodesinpathsM = Table[snodesinpaths[i, j], {i, 1}, {j, 1}];

slinksinpathM = Table[slinksinpath[i, j], {i, 1}, {j, 1}];

Selements[i_, j_] := Join[snodesinpathsM[[i, j]], slinksinpathM[[i, j]], 2];
SelementsM = Table[Selements[i, j], {i, 1}, {j, 1}];
```

VNF3 and VNF2

```
tnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathstM[[i, j, k]], 1], -1],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tlinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathstM[[i, j, k]], -1], Drop[CpathstM[[i, j, k]], 1]}],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tnodesinpathsM = Table[tnodesinpaths[i, j], {i, 1}, {j, 1}];

tlinksinpathM = Table[tlinksinpath[i, j], {i, 1}, {j, 1}];

Telements[i_, j_] := Join[tnodesinpathsM[[i, j]], tlinksinpathM[[i, j]], 2];

TelementsM = Table[Telements[i, j], {i, 1}, {j, 1}];
```

Service provider endpoints and VNF3

```
enodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathseM[[i, j, k]], 1], -1],
      {k, Length[CpathseM[[i, j]]]}]), {2}];

elinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathseM[[i, j, k]], -1], Drop[
          CpathseM[[i, j, k]], 1]}], 1], {k, Length[CpathseM[[i, j]]]}]), {2}];

enodesinpathsM = Table[enodesinpaths[i, j], {i, 1}, {j, 1}];

elinksinpathM = Table[elinksinpath[i, j], {i, 1}, {j, 1}];

Eelements[i_, j_] := Join[enodesinpathsM[[i, j]], elinksinpathM[[i, j]], 2];

EelementsM = Table[Eelements[i, j], {i, 1}, {j, 1}];
```

Endpoints connect to MANO

Service provider endpoint and MANO

```
dmanonodesinpaths[i_, j_] := Map[n# &,
    (Table[Drop[SpathsM[[i, j, k]], 1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      l_{First[#],Last[#]}, l_{Last[#],First[#]}] &, (Table[
      Drop[Transpose[{Drop[SpathsM[[i, j, k]], -1], Drop[SpathsM[[i, j, k]], 1]}],
       1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanonodesinpathsM =
  Table[dmanonodesinpaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];

dmanolinksinpathM =
  Table[dmanolinksinpath[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];

Dmanoelements[i_, j_] :=
  Join[dmanonodesinpathsM[[i, j]], dmanolinksinpathM [[i, j]], 2];

DmanoelementsM = Table[Dmanoelements[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

User endpoint and MANO

```
omanonodesinpaths[i_, j_] := Map[n# &,
    (Table[Drop[UpathsM[[i, j, k]], 1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[UpathsM[[i, j, k]], -1],
         Drop[UpathsM[[i, j, k]], 1]}], 1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanonodesinpathsM = Table[omanonodesinpaths[i, j], {i, 1}, {j, 1, NoOfMANO}];

omanolinksinpathM = Table[omanolinksinpath[i, j], {i, 1}, {j, 1, NoOfMANO}];

Omanoelements[i_, j_] :=
  Join[omanonodesinpathsM[[i, j]], omanolinksinpathM [[i, j]], 2];

OmanoelementsM = Table[Omanoelements[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNFS connect to MANO

VNF1 TO MANO

```
vnff[i_, j_] := Map[n# &,
    (Table[Drop[VpathsfM[[i, j, k]], 1], {k, Length[VpathsfM[[i, j]]]}]), {2}];

linksinf[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathsfM[[i, j, k]], -1], Drop[VpathsfM[[i, j, k]], 1]}],
     {k, Length[VpathsfM[[i, j]]]}]), {2}];

vnffM = Table[vnff[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksinfM = Table[linksinf[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementsf[i_, j_] := Join[vnffM[[i, j]], linksinfM[[i, j]], 2];

VelementsfM = Table[Velementsf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF2 TO MANO

```
vnfs[i_, j_] := Map[n# &,
    (Table[Drop[VpathssM[[i, j, k]], 1], {k, Length[VpathssM[[i, j]]]}]), {2}];

linksins[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathssM[[i, j, k]], -1], Drop[VpathssM[[i, j, k]], 1]}],
     {k, Length[VpathssM[[i, j]]]}]), {2}];

vnfsM = Table[vnfs[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksinsM = Table[linksins[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementss[i_, j_] := Join[vnfsM[[i, j]], linksinsM[[i, j]], 2];

VelementssM = Table[Velementss[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF3 TO MANO

```
vnft[i_, j_] := Map[n# &,
   (Table[Drop[VpathstM[[i, j, k]], 1], {k, Length[VpathstM[[i, j]]]}]), {2}];

linksint[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathstM[[i, j, k]], -1], Drop[VpathstM[[i, j, k]], 1]}],
     {k, Length[VpathstM[[i, j]]]}]), {2}];

vnftM = Table[vnft[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksintM = Table[linksint[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementst[i_, j_] := Join[vnftM[[i, j]], linksintM[[i, j]], 2];

VelementstM = Table[Velementst[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

The structure function and its analysis

convers the sets of elements providing a working path and control to a Boolean expression for reduction & analysis.

Service chain .

VNF1 and user endpoint.

```
Φf[i_, j_] := Table[{FelementsM[[i, j, k]] /. List → And},
   {k, Length[FelementsM[[i, j]]]}] /. List → Or

ΦfM = Table[Φf[i, j], {i, 1}, {j, 1}];

ΦfT = Table[{ΦfM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF2 and VNF1

```
Φs[i_, j_] := Table[{SelementsM[[i, j, k]] /. List → And},
   {k, Length[SelementsM[[i, j]]]}] /. List → Or

ΦsM = Table[Φs[i, j], {i, 1}, {j, 1}];

ΦsT = Table[{ΦsM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF3 and VNF2

```
Φt[i_, j_] := Table[{TelementsM[[i, j, k]] /. List → And},
   {k, Length[TelementsM[[i, j]]]}] /. List → Or

ΦtM = Table[Φt[i, j], {i, 1}, {j, 1}];

ΦtT = Table[{ΦtM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF3 and endpoint service provider

```
Φe[i_, j_] := Table[{EelementsM[[i, j, k]] /. List → And},
   {k, Length[EelementsM[[i, j]]]}] /. List → Or

ΦeM = Table[Φt[i, j], {i, 1}, {j, 1}];

ΦeT = Table[{ΦeM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

Endpoints connect to MANO

MANO and endpoint service provider

```
Φd[i_, j_] := Table[{DmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[DmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦdM = Table[Φd[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
ΦdT = Table[{ΦdM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

MANO and endpoint service provider

```
Φo[i_, j_] := Table[{OmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[OmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦoM = Table[Φo[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
ΦoT = Table[{ΦoM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNFs-> MANO

VNF1 TO MANO

```
Φvf[i_, j_] := Table[{VelementsfM[[i, j, k]] /. List → And},
   {k, Length[VelementsfM[[i, j]]]}] /. List → Or
```

```
ΦvfM = Table[Φvf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvfT = Table[{ΦvfM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF2 TO MANO

```
Φvs[i_, j_] := Table[{VelementssM[[i, j, k]] /. List → And},
   {k, Length[VelementssM[[i, j]]]}] /. List → Or
```

```
ΦvsM = Table[Φvs[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvsT = Table[{ΦvsM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF3 TO MANO

```
Φvt[i_, j_] := Table[{VelementstM[[i, j, k]] /. List → And},
   {k, Length[VelementstM[[i, j]]]}] /. List → Or
```

```
ΦvtM = Table[Φvt[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvtT = Table[{ΦvtM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

```
Φnfv = ΦfT && ΦsT && ΦeT && ΦtT && ΦvfT && ΦdT && ΦoT && ΦvsT && ΦvtT // Simplify;
```

Extracting the minimum paths and cut

The minimum path and cut sets are extracted from the above logical expressions by replacing the heads of the expressions. The And and Or head are replaced by List heads. The not operation are replaced by identity and removed. Be aware that the operation may twist the meaning unless the logical expand produces results on standard form.

Functions

```
MinPaths[s_] := BooleanConvert[s] /. {And -> List, Or -> List}
```

```
MinCuts[s_] :=
 BooleanConvert[s, "CNF"] /. {And -> List, Or -> List, Not -> Identity}
```

NFV

```
NFVminpath = MinPaths[Φnfv];
```

```
NFVmincut = MinCuts[Φnfv];
```

```
Export["D://NTNU 2 year//Master thesis//structure
    analysis//data//NFVmincut_norway_6e.mx", NFVmincut, "MX"]
```

```
D://NTNU 2 year//Master
  thesis//structure analysis//data//NFVmincut_norway_6e.mx
```

Cardinality cut sets.

```
Scenario6 = Length /@ (If[Head[#] =!= List, {#}, #] & /@ NFVmincut);
```

```
BinCounts[%, {1, Max[%] + 1, 1}]
```

```
{4, 7, 7, 121, 323, 628, 1159, 1492, 1537, 1225, 898, 476, 262, 83, 48, 9, 5}
```

```
Position[Scenario6, _?(# < 2 &)]
```

```
{{2994}, {2995}, {7894}, {8092}}
```

```
Position[Scenario6, _?(2 < # < 4 &)]
```

```
{{1}, {404}, {1857}, {3564}, {3578}, {3763}, {4456}}
```

Appendix G

Structural analysis script of Scenario 6 in Mathematica

Structure model of Scenario 6

Define network

```
Primary = Graph[{TRD ⟷ OSL1, TRD ⟷ OSL2,
     TRD ⟷ BRG, TRD ⟷ STV, OSL1 ⟷ OSL2, OSL1 ⟷ STV, STV ⟷ BRG}];
Secondary = Graph[{TRDb ⟷ OSL2b, TRDb ⟷ BRGb, TRDb ⟷ STVb,
     OSL1b ⟷ OSL2b, OSL1b ⟷ STVb, STVb ⟷ BRGb}];
ConnectionPS = Graph[{TRD ⟷ TRDb, OSL1 ⟷ OSL1b,
     OSL2 ⟷ OSL2b, BRG ⟷ BRGb, STV ⟷ STVb}];
network = GraphUnion[Primary, Secondary, ConnectionPS, VertexLabels → "Name"];

datacenter1 = Graph[{DC1 ⟷ TRD, DC1 ⟷ TRDb}, VertexLabels → "Name"];
datacenter2 = Graph[
    {OSL2 ⟷ DC2, OSL1 ⟷ DC2, OSL1b ⟷ DC2, OSL2b ⟷ DC2}, VertexLabels → "Name"];

datacenter3 =
  Graph[{DC3 ⟷ BRG, DC3 ⟷ BRGb, DC3 ⟷ STV, DC3 ⟷ STVb}, VertexLabels → "Name"];

vnf1 = Graph[{VNF1 ⟷ DC1}, VertexLabels → "Name"];

vnf2 = Graph[{VNF2 ⟷ DC1, VNF2 ⟷ DC2}, VertexLabels → "Name"];

vnf3 = Graph[{VNF3 ⟷ DC3, VNF3 ⟷ DC2}, VertexLabels → "Name"];

mano2 = Graph[{TRD → MANO2, TRDb → MANO2}, VertexLabels → "Name"];

mano1 = Graph[{BRG → MANO1, BRGb → MANO1}, VertexLabels → "Name"];

Datacenter =
  GraphUnion[network, datacenter1, datacenter3, VertexShapeFunction → "Name"];

NFV = GraphUnion[network, datacenter1, datacenter3,
    vnf1, vnf2, vnf3, mano1, mano2, VertexShapeFunction → "Name"];

NFV1 = GraphUnion[network, datacenter1,
    vnf1, mano1, mano2, VertexShapeFunction → "Name"];

NFV2 = GraphUnion[network, datacenter1,
    datacenter2, vnf2, mano1, mano2, VertexShapeFunction → "Name"];

NFV3 = GraphUnion[network, datacenter2,
    datacenter3, vnf3, mano1, mano2, VertexShapeFunction → "Name"];
```

Connect to endpoints.

```
endpoint1 = Graph[{O1 → TRD, O1 → TRDb}, VertexLabels → "Name"];

endpoint2 = Graph[{D1 → STV, D1 → STVb}, VertexLabels → "Name"];

box1 = GraphUnion[network, endpoint2, datacenter1, vnf1, VertexLabels → "Name"];

box3 = GraphUnion[network, datacenter1,
    datacenter2, vnf2, vnf1, VertexLabels → "Name"];

box2 = GraphUnion[network, datacenter1,
    datacenter2, datacenter3, vnf2, vnf3, VertexLabels → "Name"];
```

```
box4 = GraphUnion[network, datacenter2,
    datacenter3, vnf3, endpoint1, VertexLabels → "Name"];
```

```
box5 =
  GraphUnion[network, endpoint1, mano1, endpoint2, mano2, VertexLabels → "Name"];
```

Network with endpoints only.

```
endpointsnetwork = GraphUnion[Datacenter, endpoint1, endpoint2];
```

```
HighlightGraph[endpointsnetwork,
  {network, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Network with endpoints and NFV elements.

```
overallNetwork = GraphUnion[NFV, endpoint1, endpoint2];
```

```
HighlightGraph[overallNetwork, {network, mano1, mano2,
    vnf1, vnf2, vnf3, endpoint1, endpoint2}, VertexLabels → "Name"];
```

Find the path

Endpoints

```
Org[i_] := ToExpression["O" <> ToString[i]];
```

```
Dest[i_] := ToExpression["D" <> ToString[i]];
```

Datacenters

```
box[i_] := ToExpression["DC" <> ToString[i]];
```

VNFs

```
Vir[i_] := ToExpression["VNF" <> ToString[i]];
```

MANO

```
Ctrl[i_] := ToExpression["MANO" <> ToString[i]];
```

Paths between endpoints(service provider) to endpoints(user) include the service function chain.

```
Cpathsf[i_, j_] := FindPath[box1, Dest[j], Vir[1], Infinity, All]
```

```
Cpathss[i_, j_] := FindPath[box3, Vir[2], Vir[1], Infinity, All]
```

```
Cpathst[i_, j_] := FindPath[box2, Vir[3], Vir[2], Infinity, All]
```

```
Cpathse[i_, j_] := FindPath[box4, Org[i], Vir[3], Infinity, All]
```

Paths between endpoints and MANO

```
Spaths[i_, j_] := FindPath[box5, Org[i], Ctrl[j], Infinity, All];
```

```
Upaths[i_, j_] := FindPath[box5, Dest[i], Ctrl[j], Infinity, All];
```

Paths between VNFs and MANO

```
Vpathsf[i_, j_] := FindPath[NFV1, Vir[1], Ctrl[j], Infinity, All];
```

```
Vpathss[i_, j_] := FindPath[NFV2, Vir[2], Ctrl[j], Infinity, All];
```

```
Vpathst[i_, j_] := FindPath[NFV3, Vir[3], Ctrl[j], Infinity, All];
```

Define parameters.

```
NoOfPoP = 1;  NoOfVNF = 3; NoOfMANO = 2;
```

```
CpathsfM = Table[Cpathsf[i, j], {i, 1}, {j, 1, NoOfPoP}];
```

```
CpathssM = Table[Cpathss[i, j], {i, 1}, {j, 1}];
```

```
CpathstM = Table[Cpathst[i, j], {i, 1}, {j, 1}];
```

```
CpathseM = Table[Cpathse[i, j], {i, 1}, {j, 1}];
```

```
SpathsM = Table[Spaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
UpathsM = Table[Upaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
VpathsfM = Table[Vpathsf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
VpathssM = Table[Vpathss[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
VpathstM = Table[Vpathst[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

Converting paths into network elements

```
VMap = {TRD, TRDb, OSL1, OSL1b, OSL2, OSL2b, STV,
   STVb, BRG, BRGb, DC1, DC3, DC2, MANO1, MANO2, VNF1, VNF2, VNF3};
```

Service chain .

User endpoint and VNF1.

```
fnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathsfM[[i, j, k]], 1], -1],
     {k, Length[CpathsfM[[i, j]]]}]), {2}];
```

```
flinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathsfM[[i, j, k]], -1], Drop[
          CpathsfM[[i, j, k]], 1]}], 1], {k, Length[CpathsfM[[i, j]]]}]), {2}];
```

```
fnodesinpathsM = Table[fnodesinpaths[i, j], {i, 1}, {j, 1}];
```

```
flinksinpathM = Table[flinksinpath[i, j], {i, 1}, {j, 1}];
```

```
Felements[i_, j_] := Join[fnodesinpathsM[[i, j]], flinksinpathM[[i, j]], 2];
```

```
FelementsM = Table[Felements[i, j], {i, 1}, {j, 1}];
```

VNF2 and VNF1

```
snodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathssM[[i, j, k]], 1], -1],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
slinksinpath[i_, j_] := Map[If[Position[VMap, First[#]][[1, 1]] <
       Position[VMap, Last[#]][[1, 1]], Subscript[l, First[#], Last[#]],
      Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathssM[[i, j, k]], -1], Drop[CpathssM[[i, j, k]], 1]}],
      {k, Length[CpathssM[[i, j]]]}]), {2}];
snodesinpathsM = Table[snodesinpaths[i, j], {i, 1}, {j, 1}];

slinksinpathM = Table[slinksinpath[i, j], {i, 1}, {j, 1}];

Selements[i_, j_] := Join[snodesinpathsM[[i, j]], slinksinpathM[[i, j]], 2];
SelementsM = Table[Selements[i, j], {i, 1}, {j, 1}];
```

VNF3 and VNF2

```
tnodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathstM[[i, j, k]], 1], -1],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tlinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[CpathstM[[i, j, k]], -1], Drop[CpathstM[[i, j, k]], 1]}],
      {k, Length[CpathstM[[i, j]]]}]), {2}];

tnodesinpathsM = Table[tnodesinpaths[i, j], {i, 1}, {j, 1}];

tlinksinpathM = Table[tlinksinpath[i, j], {i, 1}, {j, 1}];

Telements[i_, j_] := Join[tnodesinpathsM[[i, j]], tlinksinpathM[[i, j]], 2];

TelementsM = Table[Telements[i, j], {i, 1}, {j, 1}];
```

Service provider endpoints and VNF3

```
enodesinpaths[i_, j_] := Map[n# &, (Table[Drop[Drop[CpathseM[[i, j, k]], 1], -1],
      {k, Length[CpathseM[[i, j]]]}]), {2}];

elinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
    (Table[Drop[Transpose[{Drop[CpathseM[[i, j, k]], -1], Drop[
          CpathseM[[i, j, k]], 1]}], 1], {k, Length[CpathseM[[i, j]]]}]), {2}];

enodesinpathsM = Table[enodesinpaths[i, j], {i, 1}, {j, 1}];

elinksinpathM = Table[elinksinpath[i, j], {i, 1}, {j, 1}];

Eelements[i_, j_] := Join[enodesinpathsM[[i, j]], elinksinpathM[[i, j]], 2];

EelementsM = Table[Eelements[i, j], {i, 1}, {j, 1}];
```

Endpoints connect to MANO

Service provider endpoint and MANO

```
dmanonodesinpaths[i_, j_] := Map[n# &,
    (Table[Drop[SpathsM[[i, j, k]], 1], {k, Length[SpathsM[[i, j]]]}]), {2}];
```

```
dmanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     l_{First[#],Last[#]}, l_{Last[#],First[#]}] &, (Table[
     Drop[Transpose[{Drop[SpathsM[[i, j, k]], -1], Drop[SpathsM[[i, j, k]], 1]}],
      1], {k, Length[SpathsM[[i, j]]]}]), {2}];

dmanonodesinpathsM =
  Table[dmanonodesinpaths[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];

dmanolinksinpathM =
  Table[dmanolinksinpath[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];

Dmanoelements[i_, j_] :=
  Join[dmanonodesinpathsM[[i, j]], dmanolinksinpathM [[i, j]], 2];

DmanoelementsM = Table[Dmanoelements[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

User endpoint and MANO

```
omanonodesinpaths[i_, j_] := Map[n_# &,
   (Table[Drop[UpathsM[[i, j, k]], 1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanolinksinpath[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &,
   (Table[Drop[Transpose[{Drop[UpathsM[[i, j, k]], -1],
        Drop[UpathsM[[i, j, k]], 1]}], 1], {k, Length[UpathsM[[i, j]]]}]), {2}];

omanonodesinpathsM = Table[omanonodesinpaths[i, j], {i, 1}, {j, 1, NoOfMANO}];

omanolinksinpathM = Table[omanolinksinpath[i, j], {i, 1}, {j, 1, NoOfMANO}];

Omanoelements[i_, j_] :=
  Join[omanonodesinpathsM[[i, j]], omanolinksinpathM [[i, j]], 2];

OmanoelementsM = Table[Omanoelements[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNFS connect to MANO

VNF1 TO MANO

```
vnff[i_, j_] := Map[n_# &,
   (Table[Drop[VpathsfM[[i, j, k]], 1], {k, Length[VpathsfM[[i, j]]]}]), {2}];

linksinf[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
     Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
     Transpose[{Drop[VpathsfM[[i, j, k]], -1], Drop[VpathsfM[[i, j, k]], 1]}],
     {k, Length[VpathsfM[[i, j]]]}]), {2}];

vnffM = Table[vnff[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksinfM = Table[linksinf[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementsf[i_, j_] := Join[vnffM[[i, j]], linksinfM[[i, j]], 2];

VelementsfM = Table[Velementsf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF2 TO MANO

```
vnfs[i_, j_] := Map[n# &,
    (Table[Drop[VpathssM[[i, j, k]], 1], {k, Length[VpathssM[[i, j]]]}]), {2}];

linksins[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[VpathssM[[i, j, k]], -1], Drop[VpathssM[[i, j, k]], 1]}],
      {k, Length[VpathssM[[i, j]]]}]), {2}];

vnfsM = Table[vnfs[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksinsM = Table[linksins[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementss[i_, j_] := Join[vnfsM[[i, j]], linksinsM[[i, j]], 2];

VelementssM = Table[Velementss[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

VNF3 TO MANO

```
vnft[i_, j_] := Map[n# &,
    (Table[Drop[VpathstM[[i, j, k]], 1], {k, Length[VpathstM[[i, j]]]}]), {2}];

linksint[i_, j_] :=
  Map[If[Position[VMap, First[#]][[1, 1]] < Position[VMap, Last[#]][[1, 1]],
      Subscript[l, First[#], Last[#]], Subscript[l, Last[#], First[#]]] &, (Table[
      Transpose[{Drop[VpathstM[[i, j, k]], -1], Drop[VpathstM[[i, j, k]], 1]}],
      {k, Length[VpathstM[[i, j]]]}]), {2}];

vnftM = Table[vnft[i, j], {i, 1, 1}, {j, 1, NoOfMANO}];

linksintM = Table[linksint[i, j], {i, 1}, {j, 1, NoOfMANO}];

Velementst[i_, j_] := Join[vnftM[[i, j]], linksintM[[i, j]], 2];

VelementstM = Table[Velementst[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

The structure function and its analysis

convers the sets of elements providing a working path and control to a Boolean expression for reduction & analysis.

Service chain .

VNF1 and user endpoint.

```
Φf[i_, j_] := Table[{FelementsM[[i, j, k]] /. List → And},
    {k, Length[FelementsM[[i, j]]]}] /. List → Or

ΦfM = Table[Φf[i, j], {i, 1}, {j, 1}];

ΦfT = Table[{ΦfM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF2 and VNF1

```
Φs[i_, j_] := Table[{SelementsM[[i, j, k]] /. List → And},
    {k, Length[SelementsM[[i, j]]]}] /. List → Or

ΦsM = Table[Φs[i, j], {i, 1}, {j, 1}];
```

```
ΦsT = Table[{ΦsM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF3 and VNF2

```
Φt[i_, j_] := Table[{TelementsM[[i, j, k]] /. List → And},
   {k, Length[TelementsM[[i, j]]]}] /. List → Or
```

```
ΦtM = Table[Φt[i, j], {i, 1}, {j, 1}];
```

```
ΦtT = Table[{ΦtM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNF3 and endpoint service provider

```
Φe[i_, j_] := Table[{EelementsM[[i, j, k]] /. List → And},
   {k, Length[EelementsM[[i, j]]]}] /. List → Or
```

```
ΦeM = Table[Φt[i, j], {i, 1}, {j, 1}];
```

```
ΦeT = Table[{ΦeM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

Endpoints connect to MANO

MANO and endpoint service provider

```
Φd[i_, j_] := Table[{DmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[DmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦdM = Table[Φd[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
ΦdT = Table[{ΦdM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

MANO and endpoint service provider

```
Φo[i_, j_] := Table[{OmanoelementsM[[i, j, k]] /. List → And},
   {k, Length[OmanoelementsM[[i, j]]]}] /. List → Or
```

```
ΦoM = Table[Φo[i, j], {i, 1, NoOfPoP}, {j, 1, NoOfMANO}];
```

```
ΦoT = Table[{ΦoM[[k]] /. List → Or}, {k, NoOfPoP}] /. List → And;
```

VNFs-> MANO

VNF1 TO MANO

```
Φvf[i_, j_] := Table[{VelementsfM[[i, j, k]] /. List → And},
   {k, Length[VelementsfM[[i, j]]]}] /. List → Or
```

```
ΦvfM = Table[Φvf[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvfT = Table[{ΦvfM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF2 TO MANO

```
Φvs[i_, j_] := Table[{VelementssM[[i, j, k]] /. List → And},
   {k, Length[VelementssM[[i, j]]]}] /. List → Or
```

```
ΦvsM = Table[Φvs[i, j], {i, 1}, {j, 1, NoOfMANO}];
```

```
ΦvsT = Table[{ΦvsM[[k]] /. List → Or}, {k, 1}] /. List → And;
```

VNF3 TO MANO

```
Φvt[i_, j_] := Table[{VelementstM[[i, j, k]] /. List → And},
   {k, Length[VelementstM[[i, j]]]}] /. List → Or
```

```
ΦvtM = Table[Φvt[i, j], {i, 1}, {j, 1, NoOfMANO}];

ΦvtT = Table[{ΦvtM[[k]] /. List → Or}, {k, 1}] /. List → And;


Φnfv = ΦfT && ΦsT && ΦeT && ΦtT && ΦvfT && ΦdT && ΦoT && ΦvsT && ΦvtT // Simplify;
```

Extracting the minimum paths and cut

The minimum path and cut sets are extracted from the above logical expressions by replacing the heads of the expressions. The And and Or head are replaced by List heads. The not operation are replaced by identity and removed. Be aware that the operation may twist the meaning unless the logical expand produces results on standard form.

Functions

```
MinPaths[s_] := BooleanConvert[s] /. {And -> List, Or -> List}

MinCuts[s_] :=
 BooleanConvert[s, "CNF" ] /. {And -> List, Or -> List, Not -> Identity}
```

NFV

```
NFVminpath = MinPaths[Φnfv];

NFVmincut = MinCuts[Φnfv];

Export["D://NTNU 2 year//Master thesis//structure
    analysis//data//NFVmincut_norway_7e.mx", NFVmincut, "MX"]
```

```
D://NTNU 2 year//Master
  thesis//structure analysis//data//NFVmincut_norway_7e.mx
```

Cardinality cut sets.

```
Scenario7 = Length /@ (If[Head[#] =!= List, {#}, #] & /@ NFVmincut);

BinCounts[%, {1, Max[%] + 1, 1}]
```

```
{2, 12, 7, 66, 432, 793, 1477, 2605, 3708,
 5136, 5714, 4857, 3576, 2367, 1388, 730, 344, 117, 35, 6}
```

```
Position[Scenario7, _?(# < 2 &)]
```

```
{{13 305}, {32 836}}
```

```
Position[Scenario7, _?(1 < # < 3 &)]
```

```
{{13 306}, {14 739}, {14 740}, {17 121}, {17 197}, {29 716},
 {30 864}, {31 195}, {31 525}, {32 837}, {33 115}, {33 372}}
```

Appendix H

Implementation and simulation parameters of Link SAN model in Möbius

Place Attributes:

Place Names	Initial Markings
Link_Working	1
Link_failed	0

Simulation (reward and study) parameters:

<table>
<tr><td colspan="4">Performance Variable</td></tr>
<tr><td>Reward
Function</td><td colspan="3">If (Link->Failed->Mark()==1)
{
Return 1;
}}</td></tr>
<tr><td rowspan="8">Simulator Statistics</td><td>Type</td><td colspan="2">Time Averaged Interval of Time</td></tr>
<tr><td rowspan="5">Options</td><td colspan="2">Estimate Mean</td></tr>
<tr><td colspan="2">Include Lower Bound on Interval Estimate</td></tr>
<tr><td colspan="2">Include Upper Bound on Interval Estimate</td></tr>
<tr><td colspan="2">Estimate out of Range Probabilities</td></tr>
<tr><td colspan="2">Confidence Level is Relative</td></tr>
<tr><td rowspan="2">Parameters</td><td>Start Time</td><td>0.0</td></tr>
<tr><td>Stop Time</td><td>10000000</td></tr>
<tr><td rowspan="2">Confidence</td><td>Confidence Level</td><td>0.95</td></tr>
<tr><td>Confidence Interval</td><td>0.1</td></tr>
</table>

Appendix T

Implementation and simulation parameters of Router SAN model in Möbius

Place Attributes:

Place Names	Initial Markings
Working	1
failed_MAN	0
failed	0
spare_CHW	0
failed_CHW	0
failed_SW	0
failed_FHW	0
failed_FHWt	0

Simulation (reward and study) parameters:

Performance Variable			

Reward Function	If (Link->Failed->Mark()==1)
	{
	Return 1;
	}}		

Simulator Statistics	Type	Time Averaged Interval of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	0.0
		Stop Time	10000000
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Appendix J

Implementation and simulation parameters of Datacenter SAN model in Möbius

Place Attributes:

Place Names	Initial Markings
Working	1
DC_hardware_failure	0
DC_hypervisor_failure	0

Simulation (reward and study) parameters:

Performance Variable			

Reward Function	If (Datacenter->Working->Mark()==1)
	{
	Return 0;
	}}		

Simulator Statistics	Type	Time Averaged Interval of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	0.0
		Stop Time	10000000
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Appendix K

Implementation and simulation parameters of VNF SAN model in Möbius

Place Attributes:

Place Names	Initial Markings
Working	1
vHardware_failure	0
vHypervisor_failure	0

Simulation (reward and study) parameters:

Performance Variable			

Reward Function	If (VNF->Working->Mark()==1)
	{
	Return 0;
	}}		

Simulator Statistics	Type	Time Averaged Interval of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	0.0
		Stop Time	10000000
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Appendix J

Implementation and simulation parameters of MANO SAN model in Möbius

Place Attributes:

Place Names	Initial Markings
Working	1
Mhardware_failure	0
Msoftware_failure	0

Simulation (reward and study) parameters:

Performance Variable			

Reward Function	If (MANO->Working->Mark()==1)
	{
	Return 0;
	}}		

Simulator Statistics	Type	Time Averaged Interval of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	0.0
		Stop Time	10000000
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Appendix M

Merging two models script of Reference scenario in Mathematica

- Symbolic/Numerical Evaluation

Quantitative assement

The code inclusion exclusion is based on http://www.mathematica-journal.com/2014/07/complex-system-reliability/

```
SetG[y_List] := Apply[Times, Union @@ y]
```

Unvailability based on cut - sets

A second optional parameter is introduced to take into accout set of maximum cardinality nn. The sinc set with higher cardinality is unlikely to contribute to the numerical results, since unavailabilityes are close to zero. <<this functionality is not fully tested>>

```
UnAvail[mincut_List, nn___Integer] := Module[{n, sub, mincut2},
  mincut2 = If[Head[#] =!= List, {#}, #] & /@ mincut;
  n = If[nn > 0 && nn < Length[mincut] + 1, nn, Length[mincut]];
  sub = Range[Length[mincut]];
  Sum[(-1)^(k - 1) Total[Map[SetG[mincut2[[#]]] &, Subsets[sub, {k}]]], {k, n}]]
```

Availability comparison

NB: for computing of the unaviliability we consider the elements of the min cut set with cardinality lower than 5 and and 3 inclusion-exclusion iterations

Importing the minimum cut sets;

```
NFVmincut1 = Import["D:/NTNU 2 year/Master
      thesis/structure analysis/data/NFVmincut_norway_1e.mx"];
```

Define parameters.

U_{link}=1.0066484090*10^-4
U_{router}=6.0092114391*10^-3
$U_{datacenter}$=7.7880717891*10^-4
U_{mano}= 7.7880717891*10^-4
$l_{DC1,VNF1}$=6.5884176910*10^-4
$l_{DC1,VNF2}$=6.5884176910*10^-4
$l_{DC1,VNF3}$=6.5884176910*10^-4
$l_{VNF1,DC1}$=6.5884176910*10^-4
$l_{VNF2,DC1}$=6.5884176910*10^-4
$l_{VNF3,DC1}$=6.5884176910*10^-4

network

```
NFVUnAv1 = UnAvail[Select[NFVmincut1, Length[#] < 5 &], 3] /.
    {n_DC1 → U_datacenter, n_MANO1 → U_mano, n_ → U_router, l_DC1,VNF1 → l_DC1,VNF1,
     l_DC1,VNF2 → l_DC1,VNF12, l_DC1,VNF3 → l_DC1,VNF3, l_VNF1,DC1 → l_VNF1,DC1,
     l_VNF2,DC1 → l_VNF2,DC1, l_VNF3,DC1 → l_VNF3,DC1, l_,_ → U_link} // Simplify
```

```
0.00232688
```

```
1 - NFVUnAv1
```

```
0.997673
```

Appendix N

Merging two models script of Scenario 1 in Mathematica

- Symbolic/Numerical Evaluation

Quantitative assement

The code inclusion exclusion is based on http://www.mathematica-journal.com/2014/07/complex-system-reliability/

```
SetG[y_List] := Apply[Times, Union @@ y]
```

Unvailability based on cut - sets

A second optional parameter is introduced to take into accout set of maximum cardinality nn. The sinc set with higher cardinality is unlikely to contribute to the numerical results, since unavailabilityes are close to zero. <<this functionality is not fully tested>>

```
UnAvail[mincut_List, nn___Integer] := Module[{n, sub, mincut2},
  mincut2 = If[Head[#] =!= List, {#}, #] & /@ mincut;
  n = If[nn > 0 && nn < Length[mincut] + 1, nn, Length[mincut]];
  sub = Range[Length[mincut]];
  Sum[(-1)^(k - 1) Total[Map[SetG[mincut2[[#]]] &, Subsets[sub, {k}]]], {k, n}]]
```

Availability comparison

NB: for computing of the unaviliability we consider the elements of the min cut set with cardinality lower than 5 and and 3 inclusion-exclusion iterations

Importing the minimum cut sets;

```
NFVmincut2 = Import["D:/NTNU 2 year/Master
      thesis/structure analysis/data/NFVmincut_norway_2e.mx"];
```

Define parameters.

U_{link}=1.0066484090*10^-4
U_{router}=6.0092114391*10^-3
$U_{datacenter}$=7.7880717891*10^-4
U_{mano}= 7.7880717891*10^-4
U_{vnf}=6.5884176910*10^-4

network

```
NFVUnAv2 = UnAvail[Select[NFVmincut2, Length[#] < 5 &], 3] /. {n_DC1 → U_datacenter,
     n_DC2 → U_datacenter, l_DC1,VNF1 → U_vnf, l_DC1,VNF2 → U_vnf, l_DC2,VNF3 → U_vnf, l_VNF1,DC1 → U_vnf,
     l_VNF2,DC1 → U_vnf, l_VNF3,DC2 → U_vnf, n_MANO1 → U_mano, n_ → U_router, l_,_ → U_link} // Simplify
```

```
0.00441595
```

1 - NFVUnAv2

0.995584

Appendix O

Merging two models script of Scenario 2 in Mathematica

- Symbolic/Numerical Evaluation

Quantitative assement

The code inclusion exclusion is based on http://www.mathematica-journal.com/2014/07/complex-system-reliability/

```
SetG[y_List] := Apply[Times, Union @@ y]
```

Unvailability based on cut - sets

A second optional parameter is introduced to take into accout set of maximum cardinality nn. The sinc set with higher cardinality is unlikely to contribute to the numerical results, since unavailabilityes are close to zero. <<this functionality is not fully tested>>

```
UnAvail[mincut_List, nn___Integer] := Module[{n, sub, mincut2},
  mincut2 = If[Head[#] =!= List, {#}, #] & /@ mincut;
  n = If[nn > 0 && nn < Length[mincut] + 1, nn, Length[mincut]];
  sub = Range[Length[mincut]];
  Sum[(-1)^(k - 1) Total[Map[SetG[mincut2[[#]]] &, Subsets[sub, {k}]]], {k, n}]]
```

Availability comparison

NB: for computing of the unaviliability we consider the elements of the min cut set with cardinality lower than 5 and and 3 inclusion-exclusion iterations

Importing the minimum cut sets;

```
NFVmincut3 = Import["D:/NTNU 2 year/Master
      thesis/structure analysis/data/NFVmincut_norway_3e.mx"];
```

Define parameters.

U_{link}=1.0066484090*10^-4
U_{router}=6.0092114391*10^-3
$U_{datacenter}$=7.7880717891*10^-4
U_{mano}= 7.7880717891*10^-4
U_{vnf}=6.5884176910*10^-4

```
0.000100665

0.00600921

0.000778807

0.000778807

0.000658842
```

network

```
NFVUnAv3 = UnAvail[Select[NFVmincut3, Length[#] < 5 &], 3] /.
    {n_DC1 → U_datacenter, n_DC2 → U_datacenter, n_MANO1 → U_mano, l_DC1,VNF1 → U_vnf,
     l_DC1,VNF2 → U_vnf, l_DC2,VNF2 → U_vnf, l_DC2,VNF3 → U_vnf, l_VNF1,DC1 → U_vnf, l_VNF2,DC1 → U_vnf,
     l_VNF2,DC2 → U_vnf, l_VNF3,DC2 → U_vnf, n_ → U_router, l_,_ → U_link} // Simplify
```

```
0.00376001
```

```
1 - NFVUnAv3
```

```
0.99624
```

Appendix P

Merging two models script of Scenario 3 in Mathematica

- Symbolic/Numerical Evaluation

Quantitative assement

The code inclusion exclusion is based on http://www.mathematica-journal.com/2014/07/complex-system-reliability/

```
SetG[y_List] := Apply[Times, Union @@ y]
```

Unvailability based on cut - sets

A second optional parameter is introduced to take into accout set of maximum cardinality nn. The sinc set with higher cardinality is unlikely to contribute to the numerical results, since unavailabilityes are close to zero. <<this functionality is not fully tested>>

```
UnAvail[mincut_List, nn___Integer] := Module[{n, sub, mincut2},
  mincut2 = If[Head[#] =!= List, {#}, #] & /@ mincut;
  n = If[nn > 0 && nn < Length[mincut] + 1, nn, Length[mincut]];
  sub = Range[Length[mincut]];
  Sum[(-1)^(k - 1) Total[Map[SetG[mincut2[[#]]] &, Subsets[sub, {k}]]], {k, n}]]
```

Availability comparison

NB: for computing of the unaviliability we consider the elements of the min cut set with cardinality lower than 5 and and 3 inclusion-exclusion iterations

Importing the minimum cut sets;

```
NFVmincut4 = Import["D:/NTNU 2 year/Master
      thesis/structure analysis/data/NFVmincut_norway_4e.mx"];
```

Define parameters.

U_{link}=1.0066484090*10^-4
U_{router}=6.0092114391*10^-3
$U_{datacenter}$=7.7880717891*10^-4
U_{mano}= 7.7880717891*10^-4
U_{vnf}=6.5884176910*10^-4

network

```
NFVUnAv4 = UnAvail[Select[NFVmincut4, Length[#] < 5 &], 3] /.
    {n_DC1 → U_datacenter, n_DC2 → U_datacenter, n_DC3 → U_datacenter, n_VNF1 → U_vnf,
     n_MANO1 → U_mano, l_DC1,VNF1 → U_vnf, l_DC1,VNF2 → U_vnf, l_DC2,VNF2 → U_vnf, l_DC2,VNF3 → U_vnf,
     l_DC3,VNF3 → U_vnf, l_VNF1,DC1 → U_vnf, l_VNF2,DC1 → U_vnf, l_VNF2,DC2 → U_vnf,
     l_VNF3,DC2 → U_vnf, l_VNF3,DC3 → U_vnf, n_ → U_router, l_,_ → U_link} // Simplify
```

```
0.00232878
```

1 - NFVUnAv4

0.997671

Appendix Q

Merging two models script of Scenario 4 in Mathematica

- Symbolic/Numerical Evaluation

Quantitative assement

The code inclusion exclusion is based on http://www.mathematica-journal.com/2014/07/complex-system-reliability/

```
SetG[y_List] := Apply[Times, Union @@ y]
```

Unvailability based on cut - sets

A second optional parameter is introduced to take into accout set of maximum cardinality nn. The sinc set with higher cardinality is unlikely to contribute to the numerical results, since unavailabilityes are close to zero. <<this functionality is not fully tested>>

```
UnAvail[mincut_List, nn___Integer] := Module[{n, sub, mincut2},
  mincut2 = If[Head[#] =!= List, {#}, #] & /@ mincut;
  n = If[nn > 0 && nn < Length[mincut] + 1, nn, Length[mincut]];
  sub = Range[Length[mincut]];
  Sum[(-1)^(k - 1) Total[Map[SetG[mincut2[[#]]] &, Subsets[sub, {k}]]], {k, n}]]
```

Availability comparison

NB: for computing of the unaviliability we consider the elements of the min cut set with cardinality lower than 5 and and 3 inclusion-exclusion iterations

Importing the minimum cut sets;

```
NFVmincut5 = Import["D:/NTNU 2 year/Master
      thesis/structure analysis/data/NFVmincut_norway_5e.mx"];
```

Define parameters.

U_{link}=1.0066484090*10^-4
U_{router}=6.0092114391*10^-3
$U_{datacenter}$=7.7880717891*10^-4
U_{mano}= 7.7880717891*10^-4
U_{vnf}=6.5884176910*10^-4

network

```
NFVUnAv5 = UnAvail[Select[NFVmincut5, Length[#] < 5 &], 3] /.
    {n_DC1 → U_datacenter, n_DC2 → U_datacenter, n_MANO1 → U_mano, n_MANO2 → U_mano,
     l_DC1,VNF1 → U_vnf, l_DC2,VNF2 → U_vnf, l_DC2,VNF3 → U_vnf, l_VNF1,DC1 → U_vnf,
     l_VNF2,DC2 → U_vnf, l_VNF3,DC2 → U_vnf, n_ → U_router, l_,_ → U_link} // Simplify
```

```
0.00304645
```

1 - NFVUnAv5

0.996954

Appendix B

Merging two models script of Scenario 5 in Mathematica

- Symbolic/Numerical Evaluation

Quantitative assement

The code inclusion exclusion is based on http://www.mathematica-journal.com/2014/07/complex-system-reliability/

```
SetG[y_List] := Apply[Times, Union @@ y]
```

Unvailability based on cut - sets

A second optional parameter is introduced to take into accout set of maximum cardinality nn. The sinc set with higher cardinality is unlikely to contribute to the numerical results, since unavailabilityes are close to zero. <<this functionality is not fully tested>>

```
UnAvail[mincut_List, nn___Integer] := Module[{n, sub, mincut2},
  mincut2 = If[Head[#] =!= List, {#}, #] & /@ mincut;
  n = If[nn > 0 && nn < Length[mincut] + 1, nn, Length[mincut]];
  sub = Range[Length[mincut]];
  Sum[(-1)^(k - 1) Total[Map[SetG[mincut2[[#]]] &, Subsets[sub, {k}]]], {k, n}]]
```

Availability comparison

NB: for computing of the unaviliability we consider the elements of the min cut set with cardinality lower than 5 and and 3 inclusion-exclusion iterations

Importing the minimum cut sets;

```
NFVmincut6 = Import["D:/NTNU 2 year/Master
      thesis/structure analysis/data/NFVmincut_norway_6e.mx"];
```

Define parameters.

U_{link}=1.0066484090*10^-4
U_{router}=6.0092114391*10^-3
$U_{datacenter}$=7.7880717891*10^-4
U_{mano}= 7.7880717891*10^-4
U_{vnf}=6.5884176910*10^-4

network

```
NFVUnAv6 = UnAvail[Select[NFVmincut6, Length[#] < 5 &], 3] /.
    {n_DC1 → U_datacenter, n_DC2 → U_datacenter, n_MANO1 → U_mano, n_MANO2 → U_mano, l_DC1,VNF1 → U_vnf,
     l_DC1,VNF2 → U_vnf, l_DC2,VNF2 → U_vnf, l_DC,VNF3 → U_vnf, l_VNF1,DC1 → U_vnf, l_VNF2,DC1 → U_vnf,
     l_VNF2,DC2 → U_vnf, l_VNF3,DC2 → U_vnf, n_ → U_router, l_,_ → U_link} // Simplify
```

```
0.00238961
```

1 - NFVUnAv6

0.99761

Appendix S

Merging two models script of Scenario 6 in Mathematica

- Symbolic/Numerical Evaluation

Quantitative assement

The code inclusion exclusion is based on http://www.mathematica-journal.com/2014/07/complex-system-reliability/

```
SetG[y_List] := Apply[Times, Union @@ y]
```

Unvailability based on cut - sets

A second optional parameter is introduced to take into accout set of maximum cardinality nn. The sinc set with higher cardinality is unlikely to contribute to the numerical results, since unavailabilityes are close to zero. <<this functionality is not fully tested>>

```
UnAvail[mincut_List, nn___Integer] := Module[{n, sub, mincut2},
  mincut2 = If[Head[#] =!= List, {#}, #] & /@ mincut;
  n = If[nn > 0 && nn < Length[mincut] + 1, nn, Length[mincut]];
  sub = Range[Length[mincut]];
  Sum[(-1)^(k - 1) Total[Map[SetG[mincut2[[#]]] &, Subsets[sub, {k}]]], {k, n}]]
```

Availability comparison

NB: for computing of the unaviliability we consider the elements of the min cut set with cardinality lower than 5 and and 3 inclusion-exclusion iterations

Importing the minimum cut sets;

```
NFVmincut7 = Import["D:/NTNU 2 year/Master
      thesis/structure analysis/data/NFVmincut_norway_7e.mx"];
```

Define parameters.

U_{link}=1.0066484090*10^-4
U_{router}=6.0092114391*10^-3
$U_{datacenter}$=7.7880717891*10^-4
U_{mano}= 7.7880717891*10^-4
U_{vnf}=6.5884176910*10^-4

network

```
NFVUnAv7 = UnAvail[Select[NFVmincut7, Length[#] < 5 &], 3] /.
    {n_DC1 → U_datacenter, n_DC2 → U_datacenter, n_DC3 → U_datacenter, n_MANO1 → U_mano,
     n_MANO2 → U_mano, l_DC1,VNF1 → U_vnf, l_DC1,VNF2 → U_vnf, l_DC2,VNF2 → U_vnf, l_DC2,VNF3 → U_vnf,
     l_DC3,VNF3 → U_vnf, l_VNF1,DC1 → U_vnf, l_VNF2,DC1 → U_vnf, l_VNF2,DC2 → U_vnf,
     l_VNF3,DC2 → U_vnf, l_VNF3,DC3 → U_vnf, n_ → U_router, l_,_ → U_link} // Simplify
```

```
0.00151411
```

1 - NFVUnAv7

0.998486