Optimal and Probabilistic Resource and Capability Analysis for Network Slice as a Service

Andrea Fendt

Dissertation
for the degree of
Doctor of Natural Sciences (Dr. rer. nat.)

University of Augsburg
Department of Computer Science
Software Methodologies for Distributed Systems

May, 2021

Optimal and Probabilistic Resource and Capability Analysis for Network Slice as a Service

Supervisor:	**Prof. Dr. Bernhard Bauer** Department of Computer Science, University of Augsburg, Germany
Advisor:	**Prof. Dr. Jörg Hähner** Department of Computer Science, University of Augsburg, Germany
Technical Advisors:	**Dr. Christian Mannweiler and Lars Christoph Schmelz** Bell Labs Research, Nokia, Munich, Germany
Thesis Defense:	September 30, 2021

Abstract

Network Slice as a Service is one of the key concepts of the fifth generation of mobile networks (5G). 5G supports new use cases, like the Internet of Things (IoT), massive Machine Type Communication (mMTC) and Ultra-Reliable and Low Latency Communication (URLLC) as well as significant improvements of the conventional Mobile Broadband (MBB) use case. In addition, safety and security critical use cases move into focus. These use cases involve diverging requirements, e.g. network reliability, latency and throughput. Network virtualization and end-to-end mobile network slicing are seen as key enablers to handle those differing requirements and providing mobile network services for the various 5G use cases and between different tenants. Network slices are isolated, virtualized, end-to-end networks optimized for specific use cases. But still they share a common physical network infrastructure. Through logical separation of the network slices on a common end-to-end mobile network infrastructure, an efficient usage of the underlying physical network infrastructure provided by multiple Mobile Service Providers (MSPs) in enabled.

Due to the dynamic lifecycle of network slices there is a strong demand for efficient algorithms for the so-called Network Slice Embedding (NSE) problem. Efficient and reliable resource provisioning for Network Slicing as a Service, requires resource allocation based on a mapping of virtual network slice elements on the serving physical mobile network infrastructure. In this thesis, first of all, a formal Network Slice Instance Admission (NSIA) process is presented, based on the 3GPP standardization. This process allows to give fast feedback to a network operator or tenant on the feasibility of embedding incoming Network Slice Instance Requests (NSI-Rs). In addition, corresponding services for NSIA and feasibility checking services are defined in the context of the ETSI ZSM Reference Architecture Framework. In the main part of this work, a mathematical model for solving the NSE Problem formalized as a standardized Linear Program (LP) is presented. The presented solution provides a nearly optimal embedding. This includes the optimal subset of Network Slice Instances (NSIs) to be selected for embedding, in terms of network slice revenue and costs, and the optimal allocation of associated network slice applications, functions, services and communication links on the 5G end-to-end mobile network infrastructure. It can be used to solve the online as well as the offline NSIA problem automatically in different variants.

In particular, low latency network slices require deployment of their services and applications, including Network Functions (NFs) close to the user, i.e., at the edge of the mobile network. Since the users of those services might be widely distributed and mobile, multiple instances of the same application are required to be available on numerous distributed edge clouds. A holistic approach for tackling the problem of NSE with edge computing is provided by our so-called Multiple Application Instantiation (MAI) variant of the NSE LP solution. It is capable of determining the optimal number of application instances and their optimal deployment locations on the edge clouds, even for multiple User Equipment (UE) connectivity scenarios. In addition to that multi-path, also referred to as path-splitting, scenarios with a la-

tency sensitive objective function, which guarantees the optimal network utilization as well as minimum latency in the network slice communication, is included.
Resource uncertainty, as well as reuse and overbooking of resources guaranteed by Service Level Agreements (SLAs) are discussed in this work. There is a consensus that over-provisioning of mobile communication bands is economically infeasible and certain risk of network overload is accepted for the majority of the 5G use cases. A probabilistic variant of the NSE problem with an uncertainty-aware objective function and a resource availability confidence analysis are presented.
The evaluation shows the advantages and the suitability of the different variants of the NSE formalization, as well as its scalability and computational limits in a practical implementation.

Acknowledgments

During the research and development of this dissertation I received great support from several people.

First of all, I want to thank my supervisor Prof. Dr. Bernhard Bauer for giving me the opportunity to write this Ph.D. thesis. I want to express my great gratitude for his continued solution-oriented support in all matters and the excellent guidance throughout the whole research and writing process. I also want to express my thanks to Prof. Dr. Jörg Hähner to be the second advisor of my thesis.

Special thanks go to my superiors and colleagues at Nokia Bell Labs. Without this collaboration and funding this thesis would not have been possible. In particular, I am very grateful to Dr. Henning Sanneck, Dr. Christan Mannweiler and Lars Christoph Schmelz for their expert feedback in countless meetings.

I am very grateful to my colleagues and former colleagues at the Software Methodologies for Distributed Systems Lab and the friendly and cooperative working atmosphere. I would like to mention in particular Dr. Christoph Frenzel and Dr. Simon Lohmüller who supported me in the familiarization process and enabled me to have a quick start into the topic and make fast progress.

Last but not least, a heartfelt thank you to my friends and family who always where motivating and encouraging me during the last four years.

Contents

1 Introduction

Modern society and economics are found on ubiquitous connectivity and omnipresent information. According to the Cisco VNI Global Mobile Data Traffic Forecast for 2017-2022 [1] in 2017 already 53% of mobile connections where related to smart devices. The term smart device refers to advanced multimedia or computing devices using 3G connectivity or higher, e.g., smart phones and tablets. The proportion of smart mobile devices is expected to grow to 73% and is predicted to account for 99% of mobile data traffic by 2022. This reflects a paradigm shift in mobile communication from classic voice connections to primarily mobile data usage. This tremendous change in the utilization of mobile networks has been enabled by new and improved mobile communication technology as well as recent innovations in micro computing. On the one hand, today's mobile communication networks have a huge impact on modern society and are a huge economic driver. New and improved information and telecommunication technology, especially 5G, is a door opener for new use cases, technologies and business models.[2] On the other hand, enhanced mobile network technology and smart devices induce a vast mobile data traffic growth. Global mobile data traffic increased 17-fold between 2012 and 2017 and grew by 71%, reaching 11.5 billion gigabytes per month. Between 2017 and 2022 still a 7-fold increase in mobile data traffic is expected. This corresponds to a 46% Compound Annual Growth Rate (CAGR), resulting in a predicted global data traffic of 77.5 billion gigabytes per month by 2022, see Figure 1.1. Apart from that, the data transfer rate is projected to increase from an average of 8.7 Mbps in 2017 more than 3-fold and reach 28.5 Mbps by 2022.[1]

The 5th generation of mobile networks is designed to meet these increased mobile broadband requirements as well as enable a wide variety of novel use cases, such as the Internet of Things (IoT), the Industry 4.0., but also highly safety and security critical use cases, like autonomous driving, vehicular communication and remote surgery. The following three use case categories are primarily shaping the requirements on future 5G mobile networks:

- enhanced Mobile Broadband (eMBB)
- critical Machine Type Communication (MTC)
- massive MTC

The eMBB use case requires several gigabytes of bandwidth on demand, for instance, for augmented reality or multimedia applications. This can be achieved by combining the available licensed and unlicensed radio bands and different Radio Access Technologies (RATs) with new mmWave bands below 6 GHz. Especially the new

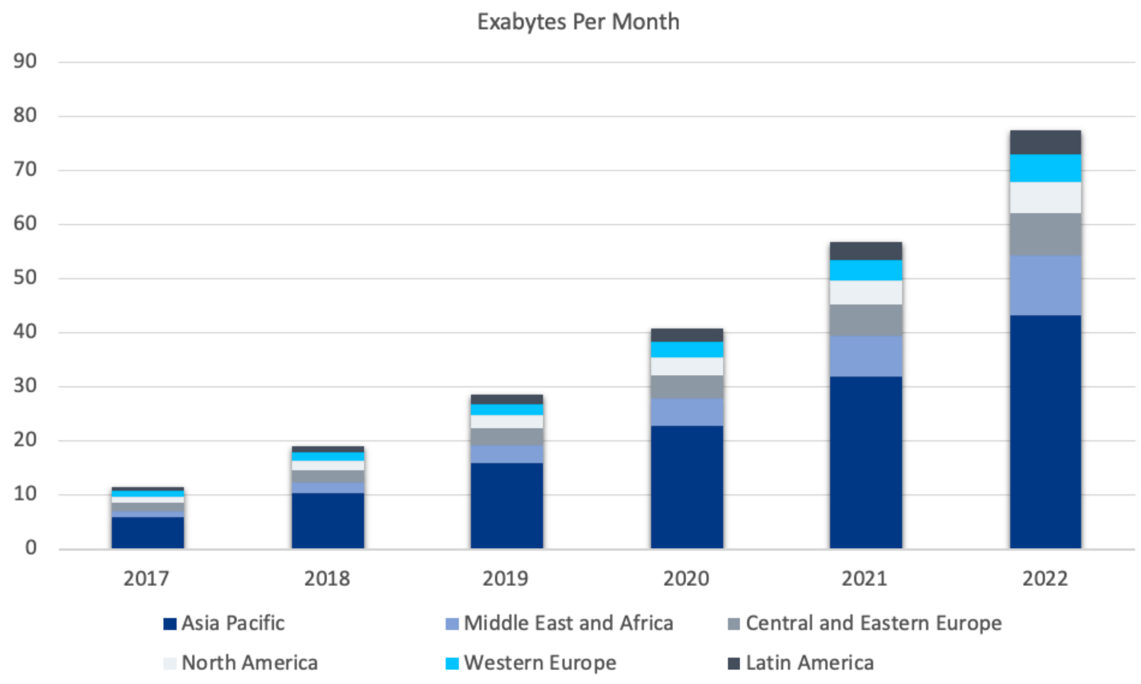


Figure 1.1: Global Mobile Data Traffic Forecast by Region [1]

mmWave bands can provide huge throughput capacities. Due to massive Multiple Input Multiple Output (MIMO), peak data rates of more than 10 Gbps will be possible in 5G. Critical MTC, like autonomous driving, often require high reliability and low latency communication. Radio link latencies below 1 ms are achieved by using shorter transmission time intervals and dynamic Time Division Duplex (TDD) in 5G. Reliability is a key design principle of the 5G network architecture as safety and business-critical use cases will be increasingly run on wireless networks. This necessitates a transformation from today's best effort mobile broadband networks to stringent and predictable service level guarantees and reliable connectivity in terms of coverage, capacity and service availability. The IoT with smart homes, smart cities and smart factories, etc., will lead to 10-100 times more devices communicating via the mobile network. This kind of Machine-to-Machine (M2M) communication typically aims at low cost and very long device battery life of up to 10 years, e.g., for smart meters.[1]
The variety of 5G use cases induces a strong diversification in mobile network requirements. Compared to today's legacy design networks, 5G must provide multiple capabilities for new Ultra-Reliable and Low Latency Communication (URLLC) and massive MTC next to eMBB use cases.[2]

In the 4G (LTE) network architecture, hard- and software are tightly coupled forming a monolithic network. This state-of-the-art legacy network architecture is not scalable and flexible enough to cope with the manifold requirements on 5G mobile networks. Although it might be possible to create a separate mobile network for each use case category, this is not efficient in terms of frequency band utilization

and therefore not economically viable. Instead, in 5G a radically different, highly flexible network architecture is deployed. Network slicing is one of its key concepts. A Network Slice Instance (NSI) is a logical, isolated, end-to-end virtual network containing all required resources and network functions to fulfill specific service requirements based on fixed Service Level Agreements (SLAs). Usually, several NSIs share the same physical infrastructure. NSIs might be instantiated, modified or terminated dynamically or on short notice.[3]
By means of network slicing, a system of systems of isolated, virtual networks sharing a common physical network infrastructure is created. The NSIs can be tailored to the specific requirements of their use cases. In order to allow to accommodate these diverse NSIs on a shared mobile network infrastructure, network functions are increasingly decoupled from their underlying hardware. This is called Network Function Virtualization (NFV). In Software Defined Networks (SDNs) virtualized network functions can be flexibly deployed on multi-purpose cloud servers, instead of dedicated devices.[2]

In current legacy networks mobile network planning and deployment of new services takes several months. This is not feasible for the 5G multi-purpose and multi-service architecture, as NSIs must be dynamically adapted to customer and user needs. When modifying an existing NSI, e.g., increasing the throughput in a specific region, or deploying a new NSI the following tasks must be completed:

Before an NSI can be deployed, its technical requirements have to be defined in detail or derived from a high-level NSI description made by the customer. Furthermore, its feasibility in the mobile network has to be analyzed and a corresponding SLA has to be negotiated between the NSI customer or tenant and the Network Slice Provider (NSP). To enable a short time to market for mobile services, fast decision making on NSI acceptance is essential. For this purpose, a 5G Network Slice Customer Portal provided by a Mobile Service Provider (MSP) is envisioned. This online customer portal allows to easily configure and order new NSIs. Tenants submit NSI requests via the portal and receive instant feedback on the feasibility of the request accompanied by an SLA and a cost estimation for the creation and operation of the requested NSI. If the NSI request is not fully realizable, downgrades might be proposed to the tenants.

A concept on how to decide whether an incoming NSI request can be accepted is necessary. To answer this question, the available resources in the network and the required resources have to be considered. Several NSIs may have been instantiated and are already running in the mobile network. Therefore, reliable resource demand predictions for the operational NSIs and the requested NSI as well as resource availability predictions for the underlying network infrastructure are needed. In addition to that, efficient Virtual Network Embedding (VNE) algorithms for embedding virtual end-to-end NSIs into substrates with both wired and wireless network elements are required.

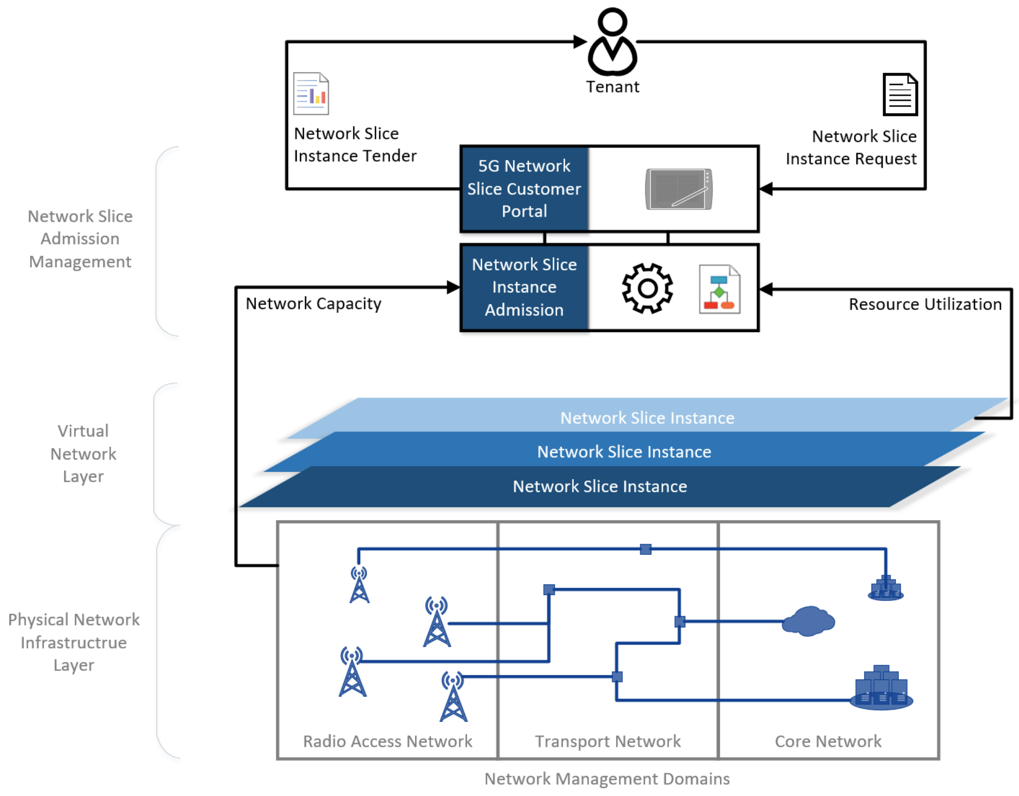

Tenant
Network Slice Instance Tender
Network Slice Instance Request
5G Network Slice Customer Portal
Network Slice Instance Admission
Network Capacity
Resource Utilization
Network Slice Admission Management
Virtual Network Layer
Physical Network Infrastructrue Layer
Network Slice Instance
Network Slice Instance
Network Slice Instance
Radio Access Network
Transport Network
Core Network
Network Management Domains

Figure 1.2: Overview over the NSIA Process

Figure 1.2 gives an overview over the NSIA process presented in this thesis. A tenant intending to set up a new NSI for his or her business or change an existing NSI interacts with the NSP via the online 5G Network Slice Customer Portal. The tenant can configure and customize the NSI requirements based on predefined NSI templates for different use case categories. The customer portal is closely linked to the Network Slice Instance Admission module. The Network Slice Instance Admission Module collects and evaluates information about available network capacities and resource demands of current and future NSIs. Therefore, the operative NSIs in the Virtual Network Layer are analyzed and their future resource demand is estimated. The network capabilities are derived from the Physical Network Infrastructure Layer and the expected future resource provisioning is predicted. A profound analysis of the requested NSI resources and capabilities as well as the residual network resources and the possibilities of reconfiguration of the operational NSIs is done. Based on that, a decision on NSI admission is made. The time elapsed between receiving the NSI request via the customer portal and the decision on the NSI acceptance should take no longer than a few minutes. If the requested NSI is feasible in the physical network, the tenant receives a tender for the requested NSI, including an associated SLA. In case the requested NSI turns out to be infeasible due to missing capabilities or resources, the tenant receives a proposal of an NSI with downgraded requirements including the associated SLAs.

1.1 Problems and Challenges

This section describes problems and challenges related to the NSIA from a network infrastructure and management software vendor point of view. This thesis helps to tackle these problems in order to pave the way towards automated NSI request assessment and deployment in 5G mobile networks.

1.1.1 Network Slice Instance Admission Problem

With 5G, the mobile network service landscape is becoming much more diverse and dynamically changing as new services and applications are developed and replace older ones. The existing, highly manual processes of network and service planning and deployment are not flexible enough for the dynamic 5G networks.

Problems The existing network and service creation and deployment times involve manual tasks like network planning and network engineering activities. This implies long service deployment times of usually several months in current Long Term Evolution (LTE) mobile networks. However, network slicing requires much faster mobile network service creation and deployment times. The configuration and implementation of a new NSI should be realized within only a few minutes. In order to overcome existing manual gaps and tremendously increase service creation and deployment times in a sliced mobile network, the following problems must be solved:

- Available network resources and capabilities from different Management Domains (MDs) must be obtained and updated automatically.
- Up-to-date and accurate predictions of resource utilization for the operational, modified and new NSI in the network must be provided.
- Network capabilities and residual resources have to be determined in an automated process.
- Efficient algorithms for resource assignment to NSIs must be provided.

Challenge 1 *A substantial reduction of service and NSI feasibility analysis, slice and service creation as well as deployment times is required to be able to cope with the dynamically changing service landscape of future 5G mobile networks. Therefore, manual gaps in service feasibility analysis, creation and deployment must be closed.*

1.1.2 Network Slice Instance Embedding Problem

NSIs are virtual networks, sharing the same physical network infrastructure. They can be permanent, but often they are subject to dynamic instantiation, change and termination. Therefore, network slicing requires that network functions and services can be flexibly allocated and configured for the NSIs. Fast decision making on network resource allocation and quick network slice instantiation is required, to enable dynamic adaption of network functions, services and resource provisioning to the customers and users needs.
The best decision on NSIA is based on an optimal embedding of a set of NSIs. That

means, to select the best NSIs with respect to a specific objective function, e.g., maximizing the revenue of the service provider. In addition to that, the optimal resource assignment to the virtual NSI elements must be determined.

Problems A comprehensive mathematical model is required to solve the resource assignment problem for NSIs in a sliced 5G mobile network, also called the Network Slice Embedding (NSE) problem in the following. The NSE problem is related to the well-researched VNE problem. However, there are some major differences, that change the characteristics of the formalization and solution:

- NSIs can have a predefined lifetime.
- NSE problems can be large, i.e., consist of a large number of NSIs with many elements to be mapped onto elements of the underlying substrate. Nevertheless, the NSE algorithm should be scalable for large problem instances.
- The substrate is a mixed wired and wireless network comprising of several different MDs.
- The NSE solution should take advantage of the fact that the UE nodes are the same in the physical as well as the virtual networks.
- The NSE solution must take consumable resources as well as non-consumable capabilities, like latency, into account.
- The NSE solution must support network function chaining, several-to-one mappings, path-splitting and Mobile Edge Computing (MEC).

Challenge 2 *A formal, mathematical description of the NSE problem that supports automated resource assignment and provides the foundation for NSIA is needed. Additionally, an automated, efficient algorithm for calculating the best solution for the NSE problem is required.*

1.1.3 Resource Uncertainty and Overbooking

Mobile communication channels are based on very limited frequency ranges. However, capacity demands are increasing massively. Hence, an overprovisioning of mobile throughput resources is infeasible, non-beneficial and contradicts with user satisfaction. Beyond that, mobile network channels are subject to fluctuations in data transmission rates. Additionally, in most NSIs the peak resource consumptions is unlikely to be requested by all NSIs simultaneously. Slight overbooking of mobile communication channels seems to be unavoidable for efficient and fair resource utilization in mobile networks. However, for many non-critical use cases, careful resource overbooking on the air interface is acceptable.

Problems Agreed, fixed service guarantees for NSIs must be met, although, for instance, frequency bands in Radio Access Network (RAN) are very limited and resource overprovisioning is economically infeasible for mobile communication resources. Furthermore, spectrum efficient Radio Access Network (RAN) subnet slice isolation is challenging, since mobile data traffic as well as channel capacities are fluctuating. In addition, user mobility and data traffic are hard to predict.[4]

Due to uncertainties in resource availability as well as demand, resource overbooking, or in the worst-case, resource shortages can occur. In order to avoid SLA violations of NSIs, the existing uncertainties must be assessed in conjunction with the NSIA.

Challenge 3 *Tackle uncertainty and overbooking in mobile networks in the context of the decision on NSI feasibility and determine the best embedding minimizing the risk of SLA violation.*

1.2 Objectives, Approach and Contributions

In this section, the three main objectives of this thesis are presented. The identified objectives address the problems and challenges described in Section 1.1. In addition, the contributions of this thesis to each objective are listed.

1.2.1 Network Slice Instance Admission Management

In this thesis, we focus on efficient NSI feasibility checking. A process and network management service should be provided enabling automated and quick NSI feasibility checking. This is a contribution to the overall NSI and service creation time reduction described in Challenge 1.

Objective 1 *Develop a process and the required network management services for automated and quick NSI feasibility analysis.*

Approach To fulfill the objective, relevant standards are considered. Several extensions and implementations of the standard are made in order to enable the needed automated and quick NSI feasibility analysis by introducing new network management services for the NSI lifecycle management. The 3GPP TS 28.530 standard [5] defines the baseline of NSI management and orchestration. It provides the description of the NSI lifecycle, the roles of the involved management entities as well as the description of several use cases. The NSIA is mentioned as one of the fundamental use cases. Further details on a top-level provisioning process of an NSI are standardized in the 3GPP TS 28.531 standard [6]. Our approach utilizes the high-level service-oriented ETSI ZSM Reference Architecture, defined in the ETSI GS ZSM 002 V1.1.1 standard [7]. Moreover, it defines a high-level NSI feasibility check, which is part of the domain orchestration services.

Contributions In order to automate and speed up the NSI feasibility analysis, the following contributions are made in this thesis:

- An NSI feasibility checking process that enables automated and quick feasibility checking in end-to-end mobile networks is defined.
- The NSI feasibility checking procedure is integrated into the standardized ETSI ZSM Framework Reference Architecture [7].

1.2.2 Network Slice Instance Embedding Models and Algorithms

In response to Challenge 2, Objective 2 is set as one of the major objectives of this thesis.

Objective 2 *Develop a formalization and efficient algorithms to solve the NSE problem.*

Approach Defining a Linear Program (LP) formalization of the NSE problem allows to determine a nearly optimal embedding, with respect to a specified objective function. The NSI feasibility can easily be derived from the nearly optimal embedding. If there are not enough resources in the physical network, some NSIs can not be embedded into the network. The excluded NSIs are regarded as being infeasible in the current physical network. The optimal NSI embedding ensures that the best, i.e., the most beneficial NSIs, are selected for deployment. Furthermore, it provides a feasible and nearly optimal resource allocation for the embedded NSIs.

Contributions An LP formalization for the NSE problem, solvable with an out-of-the-box LP solver is defined, implemented and evaluated. In a first step, a basic model is developed. This first model already allows several-to-one mappings of virtual services to physical network elements. In addition to that, it takes advantage of the predefined position of the User Equipments (UEs) in the mobile network. Furthermore, consumable resources as well as latency in mixed wired and wireless networks are taken into account in this basic NSE model.
In the second step, the model is enhanced with the possibility of path-splitting as well as a latency-aware objective function and a cost and revenue objective function. Finally, the ability for MEC is integrated into the model.
The LP formalization of the NSE problem can be solved and evaluated with common LP solvers.
The LP is implemented, tested and evaluated individually for each version of the model. Especially, the runtime and acceptance rates of the embedding algorithms are determined for each model and compared with each other on a common set of randomly generated evaluation scenarios.

1.2.3 Network Slice Embedding under Uncertainty

In order to tackle uncertainty and overbooking in mobile networks and providing a stable network, see Challenge 3, robustness is considered as a key decision factor in NSIA. Hence, the following objective is derived for this thesis.

Objective 3 *Develop a model, algorithm and risk evaluation metric to determine and assess the best NSE under resource uncertainty.*

Approach The model and algorithm to tackle uncertainty in NSE is built on the LP defined in Objective 2. The formalization is extended to consider the question of how uncertainty in resource availability and demand impacts the best NSI embedding solution. The probabilities of resource availability, i.e., the confidence that resources are available on demand, are determined for the best NSI embedding. The confidence in resource availability should be analyzed by the network operator when deciding on the NSIA.

Contributions In order to achieve Objective 3, a probabilistic model for uncertain resource availabilities and demands is defined. Furthermore, the LP based NSE solution (see Objective 2) is equipped with safety buffers and an uncertainty-aware objective function. This way, a nearly optimal embedding of the most beneficial NSIs is combined with selecting the most reliable elements for the deployment of the NSIs, i.e., the best network slice embedding, regarding robustness of resource provisioning is determined. In addition, a metric is provided to determine the confidence in resource availability. The risk of SLA violation for an embedded NSI, i.e., the probability of failure to provide the guaranteed resources is analyzed in order to assess the robustness of a specific NSI embedding. If the robustness is considered good enough by the decision maker, that means, the probability that the actual required resources will be available when requested is acceptable, the NSIs can be deployed according to the proposed embedding. This can include a deliberate, careful overbooking of one or several mobile network resources.

1.3 Outline

Figure 1.3 gives an overview over the chapters of this thesis and the objectives they are associated with. The first part of this thesis introduces the topic and provides the necessary background information. The main part consists of Chapters 3 to 6, covering the four main objectives of this work. Chapter 3 provides an architecture and process that serves as a framework for the following contributions. The optimal network slice embedding, described in Chapter 4, is the foundation for the network slice embedding under uncertainty in Chapter 5. Chapter 6 provides an extensive evaluation of the presented NSE algorithms and its variants. The thesis ends with an overall conclusion and outlook for future work.

Chapter 1 Introduction motivates the topic and the key research questions of this thesis. The problems and challenges of NSIA in a sliced 5G mobile network are illustrated. Beyond that, the objectives and contributions as well as a list of the authors publications associated with this thesis are summarized.

Chapter 2 Foundations introduces the required theoretical and technological background this thesis is based on. An overview over technologies and standardization of network slicing is given. Furthermore, VNE, especially focusing on the relevant base algorithms, is introduced.

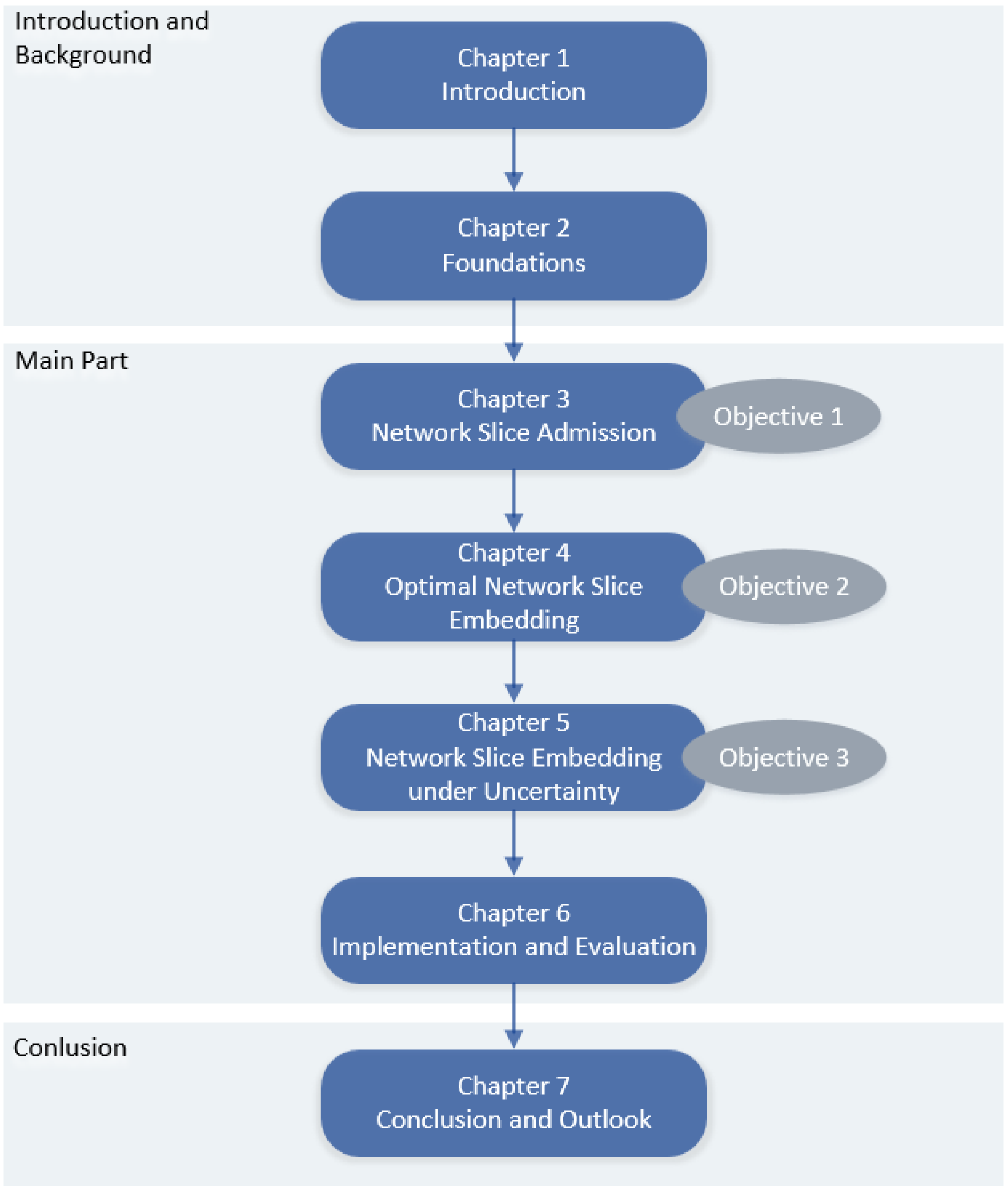

Introduction and Background
Chapter 1 Introduction
Chapter 2 Foundations
Main Part
Chapter 3 Network Slice Admission
Objective 1
Chapter 4 Optimal Network Slice Embedding
Objective 2
Chapter 5 Network Slice Embedding under Uncertainty
Objective 3
Chapter 6 Implementation and Evaluation
Conlusion
Chapter 7 Conclusion and Outlook

Figure 1.3: Outline of this Thesis

Chapter 3 Network Slice Admission presents an architecture and process for NSIA. This chapter provides a framework for the contributions in Chapters 4 to 6.

Chapter 4 Optimal Network Slice Embedding provides an LP formalization of the NSE problem, which can be solved with an out-of-the-box LP solver. First, a basic, extensible version of the NSE model is defined. In a modular approach, the model can be enhanced with further features.

Chapter 5 Network Slice Embedding under Uncertainty builds on the nearly optimal NSE algorithm, described in Chapter 4. It enhances the model by including resource uncertainty considerations and provides a method to evaluate the confidence in resource availability.

Chapter 6 Implementation and Evaluation evaluates the optimal NSE models and its variants, introduced in Chapter 4 as well as the NSE under uncertainty, introduced in Chapter 5. The efficiency of these solutions regarding, for instance, the acceptance rates and the resource availability confidences as well as the runtime and scalability are analyzed.

Chapter 7 Conclusion and Outlook summarizes the results of this thesis and draws conclusions with respect to the defined objectives. The thesis is concluded with an outlook on future research on NSI resource management in the preparation phase.

1.4 Publications

Parts of this thesis have already been published in previous scientific publications and patent applications. A list of these publications with a brief summary and their relevance for this thesis is given below. The ideas, concepts, algorithms and results published by the author of this thesis are not additionally cited throughout this thesis.

1.4.1 Scientific Publications and Patent Applications

1. Andrea Fendt, Lars Christoph Schmelz, Wieslawa Wajda, Simon Lohmüller and Bernhard Bauer. "A Network Slice Resource Allocation Process in 5G Mobile Networks" In *Innovative Mobile and Internet Services in Ubiquitous Computing* (July 2018).[8]
   In this paper, the authors provide a vision of an end-to-end NSI resource allocation process allowing to give fast feedback to a network operator or tenant on the feasibility of embedding new NSIs. The basic concepts and ideas have been developed in discussions with the co-authors.
   The author of this thesis is the main contributor of the detailed concepts and the publication itself. This work is primarily integrated in Chapter 3.

2. Andrea Fendt, Simon Lohmüller, Lars Christoph Schmelz and Bernhard Bauer. "A Network Slice Resource Allocation and Optimization Model for End-to-End Mobile Networks." In *IEEE 1st 5G World Forum (5GWF'18) Conference Proceedings* (July 2018), pp. 262-267.[9]
   In this paper, a standardized Integer Linear Program (ILP) for offline mobile network slice embedding, especially focusing on resource allocation and virtual node as well as link mapping is presented, implemented and evaluated.
   The author of this thesis is the main contributor of the developed model, solution and evaluation. The results of this paper are integrated in Chapter 4.

3. Andrea Fendt, Christian Mannweiler, Lars Christoph Schmelz and Bernhard Bauer. "A Formal Optimization Model for 5G Mobile Network Slice Resource Allocation." In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (Nov. 2018), pp. 101-106.[10]
   In this paper, a mathematical model for solving the offline Network Slice Embedding Problem formalized as a standard Mixed Integer Linear Proram (MILP) is presented. A latency sensitive objective function guarantees the optimal network utilization as well as minimum latency in the network slice communication.
   The author of this thesis is the main contributor of the developed model, solution and evaluation. The results of this paper are integrated in Chapter 4.

4. Andrea Fendt, Borislava Gajic, Christian Mannweiler and Lars Christoph Schmelz. "An Apparatus and Method for Network Slice Instance Feasibility Checking in Network Slice Instantiation.", Invention and Patent Application (Apr. 2019).[11]
   The author of this thesis, together with three fellow researchers, provides a method for quick NSI feasibility analysis and robustness evaluation in the NSI preparation phase. The ideas and concepts as well as the patent application itself have been developed and written in cooperation with the other authors. Hence, the invention is the intellectual property of all involved authors in equal parts. This work is primarily used in Chapter 3 and serves as a basis for the models in the Chapters 4 and 5.

5. Andrea Fendt, Christian Mannweiler, Lars Christoph Schmelz and Bernhard Bauer "An Efficient Model for Mobile Network Slice Embedding under Resource Uncertainty" In *2019 16th International Symposium on Wireless Communication Systems (ISWCS)* (Aug. 2019), pp. 602-606.[12]
   In this work, the authors present an uncertainty-aware model and solution for mobile NSE. The proposed solution allows an informed decision on NSIA. The decision is based on the guaranteed end-to-end mobile network resources that have to be provided on the one hand and the capacities and capabilities of the underlying network infrastructure on the other hand.

The author of this thesis is the main contributor of the developed model, solution and evaluation. The results of this paper contribute to Chapter 5.

6. Katja Ludwig, Andrea Fendt and Bernhard Bauer. "An Efficient Online Heuristic for Mobile Network Slice Embedding." In *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)* (Feb. 2020), pp. 139-143.[13]
   The paper presents a heuristic for the NSE problem, which is based on topology-aware node ranking. Three alternative ranking techniques are introduced and analyzed.
   In this paper the results of Katja Ludwig's master thesis are published. The author of this thesis supervised the master thesis and provided the fundamental ideas and concepts for the master thesis as well as the paper and supervised and reviewed the paper. The results of this paper are out of scope of this thesis.

7. Andrea Fendt, Christian Mannweiler, Katja Ludwig, Lars Christoph Schmelz and Bernhard Bauer "End-to-End Mobile Network Slice Embedding Leveraging Edge Computing" In "NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium" (April 2020), pp. 1-7.[14]
   In this work, the authors propose an ILP-based formulation and nearly optimal solution of the NSE problem leveraging edge computing. The optimal set of NSIs, in terms of revenue and cost, is determined. The presented solution provides the optimal number of application instances and their optimal deployment locations on the edge clouds, even for multiple User Equipment (UE) connectivity scenarios.
   The author of this thesis is the main contributor of the developed model, solution and evaluation. The results of this paper are used in Chapter 4.

2 Foundations

The goal of this chapter is to provide the necessary theoretical background and the used technologies and algorithms this thesis is based on. The basics presented in this chapter are not exhaustive as they focus on the required foundation and the background of this thesis.
First, management frameworks and fundamental concepts of current mobile networks are introduced especially focusing on network slicing. Subsequently, an overview over selected concepts in the prior art of VNE that have been used in Chapter 4 and Chapter 5 is presented.

2.1 Mobile Networks

In this section, the relevant foundations on mobile network management consulting the Next Generation Mobile Networks (NGMN) as well as the European Standards Organization (ETSI) Zero touch network and Service Management (ZSM) standards are introduced.

Standardization of mobile network management focuses on mobile network automation, which is essential to agile and quick service delivery on the one hand and to economic sustainability of mobile network provisioning on the other hand. The introduction of new use cases in the 5th generation of mobile networks induces a transformation in the mobile network management and orchestration. Increased demands of ubiquitous connectivity and imperceptive latency and seemingly unlimited communication capacity as well as support for massive Machine Type Communication (mMTC) require highly flexible and programmable networks.[15]

Network slicing is introduced as a new business model, reinventing the mobile service creation, orchestration and management as well as improving agility and cooperation across the different network domains. This leads to an increased complexity of mobile networks intensifying the need for full automation of service delivery. Digital, fully-automated lifecycle management systems are envisioned that include the automated service, network, cloud and resource management. The presented architectural frameworks are designed to facilitate automated closed-loop control and machine learning with no or only minor human intervention.[15]

2.1.1 NGMN Mobile Network System Description

The NGMN Alliance proposes a description of the mobile network system environment leveraging the structural separation of hardware and software in the network with the target of improving agility and automation of network management and orchestration. Therefore, network programmability is enhanced enabling flexible and dynamic as well as automated configuration and reconfiguration of the mobile network system. The proposed system has a service-based architecture design with modularized services.[16]

The NGMN 5G System Environment is illustrated in Figure 2.1. It consists of three layers, the Infrastructure Resource Layer, the Business Enablement Layer and the Business Application Layer.
The Infrastructure Resource Layer is the lowest layer. It comprises the physical resources of the mixed wired and wireless mobile network. This includes access nodes, edge and central cloud nodes, providing processing and storage resources as well as networking nodes. In addition, the 5G infrastructure provides connection to external public and private IP networks. Among others, classic UEs (like mobile phones, smart phones, tablets), IoT devices, sensors, smart homes, Industry 4.0 networks and autonomous vehicles are connected via the 5G RAT. The infrastructure resources are utilized by the higher layers of the architecture, especially the Business Enablement Layer. In addition, they are exposed via a relevant Application Programming Interfaces (APIs) to the End-to-End (E2E) Management & Orchestration entity.
The E2E Management & Orchestration includes optimization, network slice management, automation as well as self-organizing functionalities. Network slices for specified application scenarios are created, managed and decommissioned by the E2E Management & Orchestration entity. This includes the setup of function chains of relevant Network Functions (NFs) as well as performance configuration and the mapping of NFs on infrastructure resources and resource capacity management. Beyond that, the E2E Management & Orchestration entity provides interfaces for third parties to create and manage their network slices themselves.
The Business Enablement Layer provides a library of all functions in particular for converged fixed-mobile network modular architecture building blocks. The Business Enablement Layer hosts NFs including Control Plane (CP) and User Plane (UP) functions as well as a set of configuration parameters, e.g., RAT configurations, and state information functions. The CP, which is responsible for signaling traffic and routing, is strictly separated from the UP. The UP, also referred to as the data plane, is responsible for transmitting user traffic. Required aspects of these functions and capabilities are exposed through relevant APIs towards the orchestration entity and towards the Business Application Layer. The functions can be provided with different characteristics, for instance, with higher or lower performance for use cases with different user mobility characteristics.
The highest layer, the Business Application Layer comprises specific applications and services of operator, enterprise, verticals or third parties. In virtualized envi-

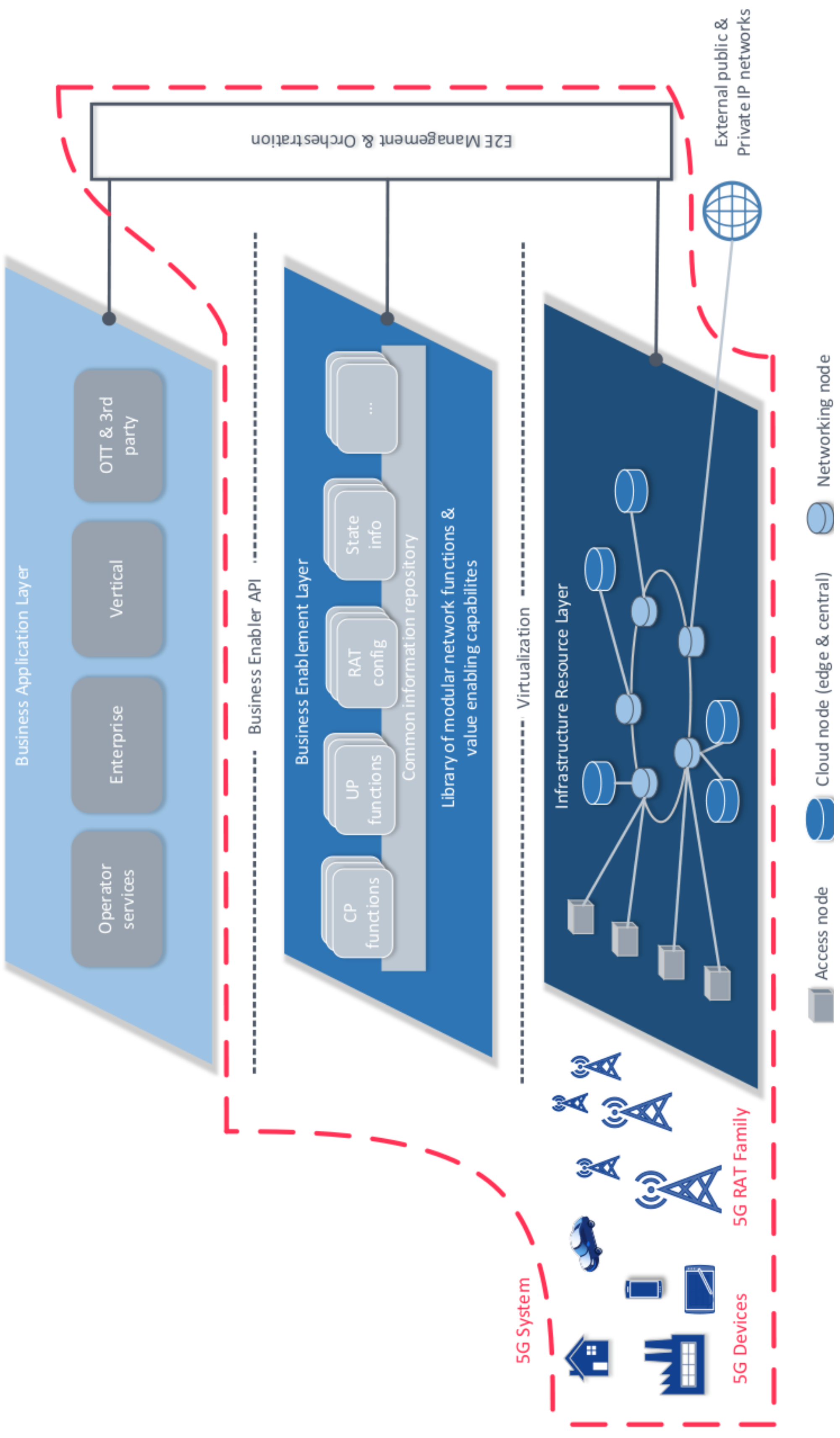

E2E Management & Orchestration
External public & Private IP networks
Business Application Layer
Operator services
Enterprise
Vertical
OTT & 3rd party
Business Enabler API
Business Enablement Layer
CP functions
UP functions
RAT config
State info
...
Common information repository
Library of modular network functions & value enabling capabilites
Virtualization
Infrastructure Resource Layer
5G System
5G RAT Family
5G Devices
Access node
Cloud node (edge & central)
Networking node

Figure 2.1: NGMN 5G System Environment [16]

ronments, those applications and services are hosted on Multi-Access Edge Computing (MAEC) hosts or datacenters. The Operator services, the so-called NSP Applications, refer to regular telecommunication services such as voice, messaging and internet access. They include NSP differentiating services offered to its own subscribers. The 5G system provides rapid instantiation, modification and removal of NSP Applications. In contrast to that, the Enterprise service applications are NSP Applications offered exclusively to their enterprise customers providing end-to-end enterprise services. Using the so-called Authorized Over-The-Top (OTT) and 3^{rd} Party Service Applications, NSPs can host services for other authorized 3^{rd} party and OTT applications. The 5G system must support instantiation requests as well as management and monitoring of service applications for OTT players and 3^{rd} parties. Specific aspects of the Business Application Layer applications and services are exposed to the E2E Management & Orchestration.[16]

2.1.2 ETSI ZSM Framework Reference Architecture

ETSI contributes to the 5G ecosystem with the Industry Specification Group (ISG) on ZSM, founded in December 2017. It defines an architectural framework for scalable full automation of service provisioning and network operation, in particular the configuration, deployment, assurance and optimization of services and network tasks and processes.
In the ZSM 002 [7] and 003 [17] the Framework Reference Architecture is provided, while the specification work is still ongoing.[15]

The proposed framework is compatible with the NGMN 5G system environment, described in Section 2.1.1 of the NGMN Alliance. Both system descriptions are service-based and modularized systems based on a physical network infrastructure layer, enabling services and applications in the layers above them. Moreover, the Framework Reference Architecture specified in the ZSM 002 and 003 standard documents complements the 3GPP specifications of Release 15. In particular, the 3GPP Technical Specification 28.533 [18] defines the management service deployment based on the ZSM framework.
The ETSI ZSM Framework Reference Architecture is especially focusing on an automated network and service management in a multi-vendor environment. Thus, the specified architecture is based on the following main design principles:

1. Modularity
2. Extensibility
3. Scalability
4. Model-driven, open interfaces
5. Closed-loop management automation
6. Support for stateless management functions
7. Resilience
8. Separation of concerns in management
9. Service composability
10. Intent-based interfaces

11. Functional abstraction
12. Simplicity
13. Designed for automation

Services in the ETSI ZSM Framework Reference Architecture are **modular**, i.e., they have a clearly defined scope, are self-contained and interact only over well-defined interfaces. Management services in the ETSI ZSM Framework Reference Architecture are defined as a mechanism offering capabilities to a service consumer via a standardized interface, a so-called management service end-point. Besides modularity, services and capabilities are **extensible**, that means, new services and capabilities can be added to the network and MDs without the need to change the design of the implementation or interaction in the existing network. Beyond that, services within the Framework Reference Architecture are required to be **scalable**, that means, they must be adaptable to increasing as well as decreasing demands. Besides this, vendor-neutral resource management is crucial in the aspired multi-vendor environment. Therefore, **model-driven, open interfaces** are another important design principle. Automation and self-optimization are facilitated by the 5th design principle of **closed-loop management automation**. Modularity, reusability and vendor-neutrality are further improved by the required **support for stateless management functions**, which results in a separation of data processing from data storage. In order to avoid complex monolithic systems, **separation of concerns** in management between domain management and end-to-end service management across domains is required. Moreover, **resilience** is an important design principle, the framework should allow degradation of infrastructure and other management services as well as return to normal operation. Service provisioning in the ETSI ZSM Framework Reference Architecture is based on modular service structures. Management services of MDs can be combined to create new management services. This is called **service composability**. Through **intent-based interfaces** vendor-neutrality and usability is improved, as it allows for an abstraction from technology and vendor-specific details used in declarative user interfaces. Similarly, the principle of **functional abstraction**, improves the usability of the system. Function abstractions is defined as generalizing the behavior of related entities. The function encapsulates different variants and so abstracts from details in different variants. Last but not least, minimizing complexity to create a network architecture that is as **simple** as possible as well as designing a system facilitating **automation** in network service management are self-evident design principles.[7]

An overview over the ETSI ZSM Framework Reference Architecture is shown in Figure 2.2. It consists of distributed management functions/services and data services of different MDs. The MDs administrate individual domain-specific resources, which can be physical network infrastructure or virtual resources. That means, the domain infrastructure resources can be Physical Network Functions (PNFs), software-based Virtualized Network Functions (VNFs) or cloud-based resources. Related management and data services, for instance, services using the same set of infrastructure resources, the same technology or services that are owned by the same infrastructure or service provider are grouped into one administrative domain. This facilitates the

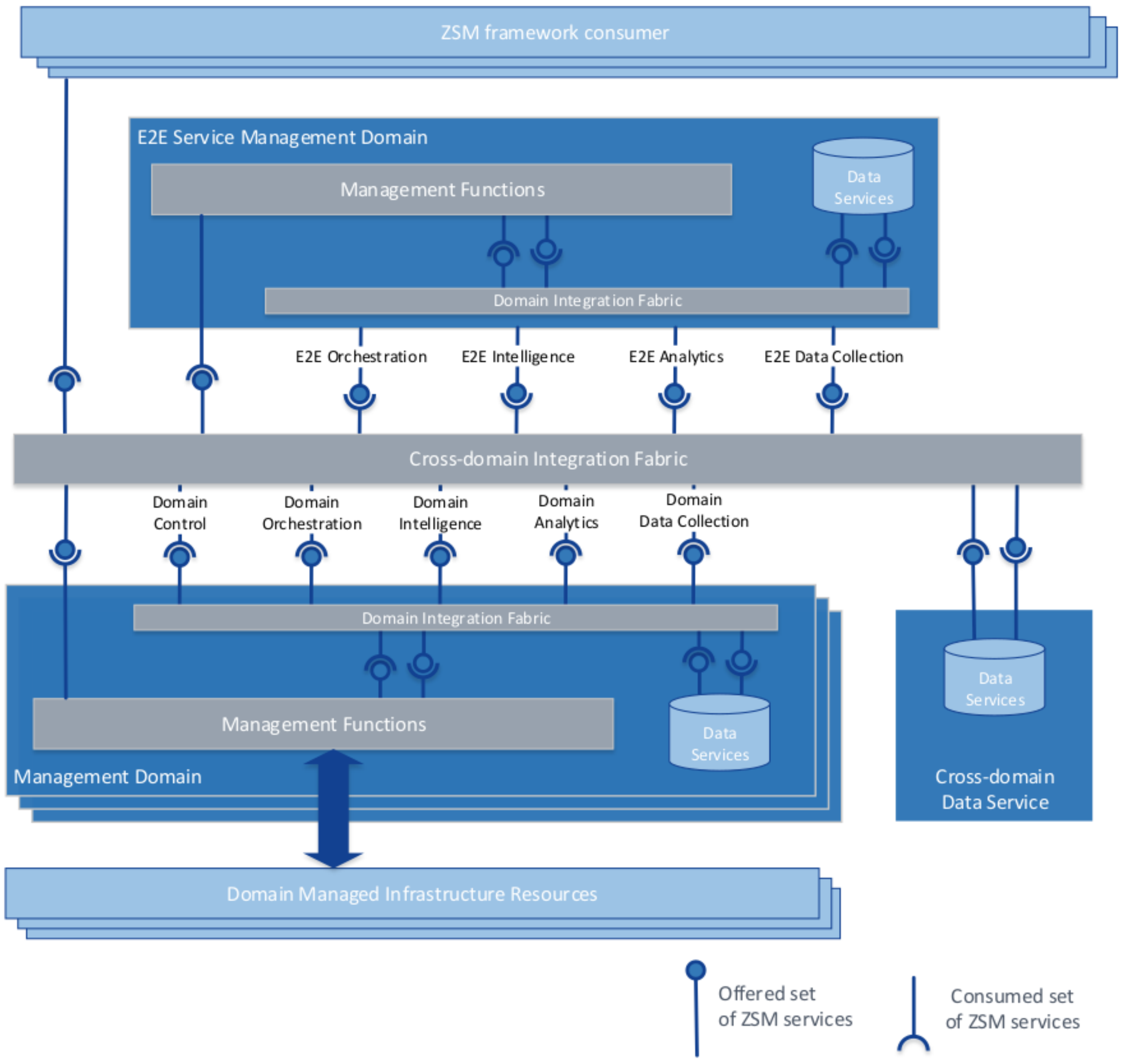


Figure 2.2: ZSM Framework Reference Architecture [7]

separation of concerns between different network domains. For instance, the RAN, the transport and the core network should be separated into different MDs.
The E2E Service Management Domain represents a special case of a MD, it is located on a higher level of abstraction. In contrast to the regular MDs, the E2E Service Management Domain does not directly access infrastructure resources, instead it manages the services provided by several MDs. Each MD, including the E2E Service Management Domain, is equipped with management functions exposing a set of ZSM management by externally visible service end-points of selected management services within the MD. ETSI defines management functions as "*entities that produce and/or consume management services.*"[7, p. 16] The ZSM framework enables the creation of loosely-coupled management function chains collectively providing end-to-end or domain-specific capabilities for automated service, network and infrastructure management. Service capabilities are uniformly offered by the management services. This enables highly automated service consumption and interaction between different MDs and the infrastructure. Management services can be combined hierarchically creating higher levels of abstractions and management services with a broader scope. Management service producers might directly interact with infrastructure resources using their management interfaces or indirectly invoke their consuming management services.
The MDs have a domain integration fabric, which contains functions controlling access as well as internal or cross-domain visibility of the management services provided by the domain. They provide dedicated data services via the domain integration fabric. Data services manage data access and provide a consistent data persistence. Selected data services can be exposed for cross-domain usage or provided to the ZSM framework consumers via the cross-domain integration fabric. This way, stateless management functions without their own data persistence can be realized.
The cross-domain integration fabric has well-defined interfaces administrating all cross-domain communication in the ZSM Framework Reference Architecture. It provides service capabilities to the accessing end-points. Beyond that, the MDs act as a consumer of ZSM service capabilities provided by the E2E Service Management Domain via the cross-domain integration fabric. The integration fabric allows cross-domain data sharing, communication, service consumption and integration as well as the integration of 3^{rd} party management systems.
Furthermore, the eE2E Service Management Domain and the MDs provide services which can be consumed by the so-called ZSM framework consumers, i.e., external service consumers. Typical ZSM framework consumers are, for instance, the digital store fronts, Business Support System (BSS) applications, web portals of other ZSM framework instances or additional user interfaces, for instance, for human users. Digital store fronts offer products, which are based on E2E services, to external customers as well as receive and handle requests from external customers for service deployment.
The E2E Service Management Domain provides the following management services, which can be provided as end-user services, towards the ZSM framework consumers:

- E2E service orchestration
- E2E service intelligence

- E2E service analytics
- E2E service data collection

These end-to-end services are composed of services in various MDs, exposed via one or more end-points within the cross-domain integration fabric. The E2E service orchestration services coordinate the configuration, provisioning and lifecycle management of customer facing end-to-end network services. While the E2E service intelligence services comprise end-to-end management services responsible for automated decision making, including, for instance, troubleshooting and fixing issues across MDs, the E2E service analytics services derive insights in the performance of the managed services and service-related Key Performance Indicators (KPIs). In addition, the E2E service data collection services gather all managed service-related data, for example performance data.[7]

2.2 Network Slicing

Based on the 5G environment and system descriptions presented in Section 2.1 in this chapter the concepts and management of network slices, as specified by the relevant telecommunication industry consortia, in particular the NGMN, 3rd Generation Partnership Project (3GPP) and Groupe Speciale Mobile Association (GSMA), are introduced.
First, the relevant definitions are introduced, then the information model and the business roles and instance management of the network slices are presented.

2.2.1 Definitions

Network slicing is one of the key concepts in 5G to enable new use cases with very diverse requirements and an increased need for ubiquitous and robust mobile communication. The NGMN Alliance was the first to introduce the term network slicing for mobile networks in their 5G white paper [19, p. 45 ff.] in February 2015. Network slices provide virtual private services in end-to-end mobile networks using isolated, logical networks running on a common mobile network infrastructure. The 3GPP specifications distinguish between network slice templates, a *"description of the structure (and contained components) and configurations of a network slice"*[3], and NSIs, defined as *"a set of network functions and the resources for these network functions which are arranged and configured, forming a complete logical network to meet certain network characteristics"*[3], i.e., network slice templates are blueprints that can be used to ease the definition of concrete instances of network slices. In contrast to that, NSIs are definitions of a concrete implementation of a specific network slice, which are complete, i.e., contain all required functionalities and resources to provide the defined communication services for their intended use case. This completeness implies that NSIs are defined end-to-end, i.e., the communication services cover all necessary parts of the network. This includes all required parts of the mobile network, in particular all necessary NFs and physical as well as virtualized resources in the access, transport and core networks as well as the corresponding clouds and servers providing the services. It is important to note that network slices

may comprise virtualized as well as non-virtualized components.
Communication services offered by a network slice can be classified into different connection types or service categories, like eMBB, URLLC and mMTC. New categories might emerge as new use cases arise. Typical examples for URLLC communication services are vehicle-to-vehicle (V2V) or vehicle-to-everything (V2X), while Virtual and Augmented Reality as well as video streaming services are in the class of eMBB. IoT or smart factory communication services are often associated with the mMTC communication service category. As a consequence, some endpoint devices and UEs are connected to only one network slice, for instance, road side sensors and endpoint devices deployed in automotive vehicles, while other devices, like smart phones or tablets, could use the services of several connected network slices simultaneously.[3, 16, 20]

2.2.2 Network Slice Meta Model

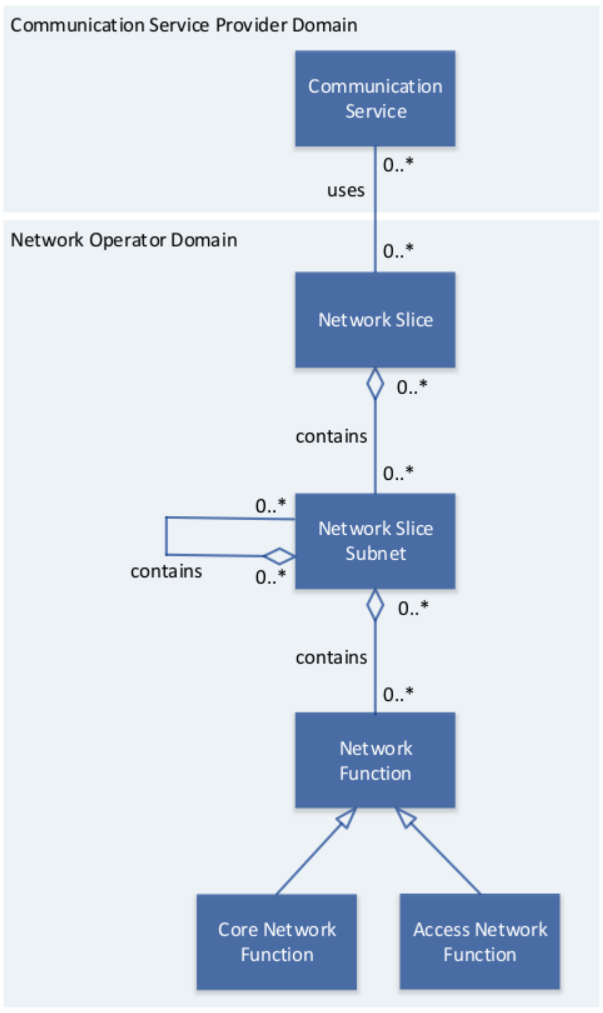


Figure 2.3: 3GPP Network Slicing Meta Model [3]

The relations between the communication services and network slices is shown in

Figure 2.3. The Figure provides a meta model for the components of network slices. Usually, a communication service is provided by a single network slice, however there are specific exceptions for service bundles provided by more than one Packet Data Unit (PDU) connectivity service. A communication service can be part of multiple different network slices. Whereas in practice, the service and network slice definition is done by the MSP. The provider can define, for instance, one network slice per service category or decide for a more fine-grained network slice definition, for instance, creating separate NSIs for different communication services or even for different network slice customers (tenants).
A network slice might contain multiple network slice subnets, which can be nested, i.e., network slice subnets can be composed of further subordinated network slice subnets. NSIs provide a complete (end-to-end) service. A Network Slice Subnet Instance (NSSI) is a logical subnetwork, comprised of configured network services, that provides only a part of the NFs and resources required for a network slice or service instance. Network slice subnet templates are used to describe the structure, components and configuration of a network slice subnet. They allow to assemble the end-to-end NSI with different NSSIs that contain, for example, the RAN or core network components, respectively.
Network slice subnets make use of multiple NFs, which can be used in several network slice subnets simultaneously. ETSI NFV 003 [21] defines an NF as a *"functional building block within a network infrastructure, which has well-defined external interfaces and a well-defined functional behaviour."*[21, p. 5] In practice, an NF often is a network node or a physical appliance [21]. NFs can be categorized into core NFs and access NFs. Typical core NFs are, for instance, the Access and Mobility Management Function (AMF), the Session Management Function (SMF), the User Plane Function (UPF) and the Policy Control Function (PCF). Examples for access NFs are Distributed Units (DUs) as well as the Central Unit - Control Planes (CU-CPs) and the Central Unit - User Plane (CU-UP) in the 3GPP RAN as well as the Base Transceiver Stations (BTSs) and the Next Generation Node Base Stations (gNBs). The term VNF refers to softwareized NFs that are not bound to dedicated hardware.[3, 16, 20, 22]

In the course of service-based mobile network architectures, NSIs are usually offered in form of predefined telecommunication services, called Network Slice as a Service (NSaaS). I.e., that an Mobile Network Operators (MNOs) offers a complete network slice, instead of a communication service, to a tenant allowing the tenant to use the network slice and deploy its own communication services as well as add new NFs to it. The tenant can acquire NSSIs to be used as building blocks to form a customized NSI based on an NSaaS. Tenants might use the obtained NSI to provide communication services to their end-customers or for their own business operations.[5]

2.2.3 Business Roles

The NSaaS paradigm involves four involved business roles specified, for instance, by the 5G NORMA Project in [23] and [24]. In Figure 2.4, the business roles of the

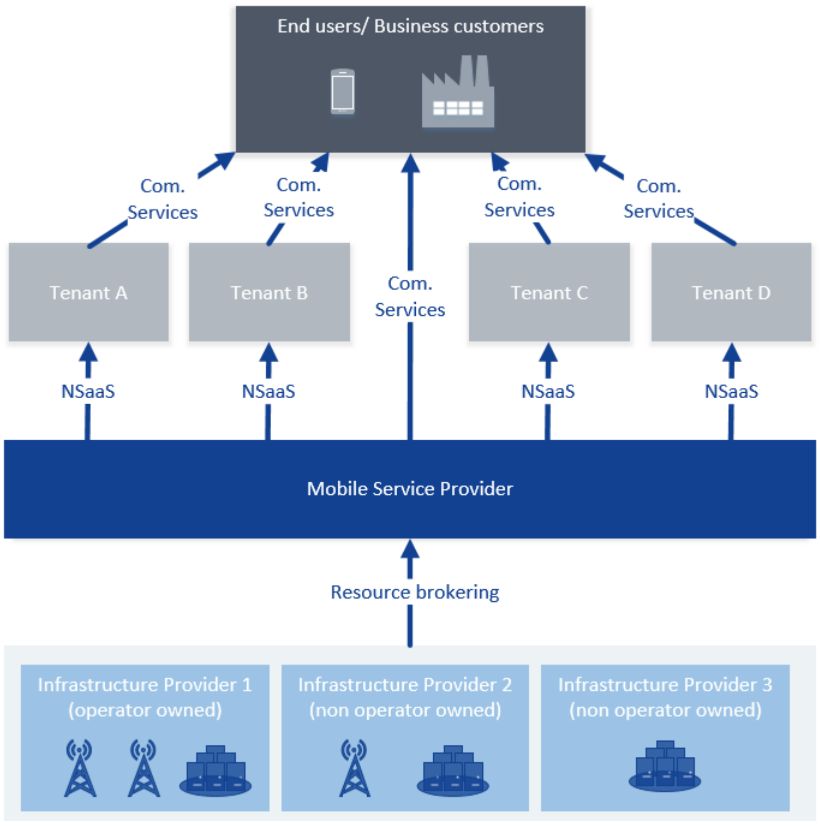

End users/ Business customers
Com. Services
Com. Services
Com. Services
Com. Services
Com. Services
Tenant A
Tenant B
Tenant C
Tenant D
NSaaS
NSaaS
NSaaS
NSaaS
Mobile Service Provider
Resource brokering
Infrastructure Provider 1 (operator owned)
Infrastructure Provider 2 (non operator owned)
Infrastructure Provider 3 (non operator owned)

Figure 2.4: High-Level Business Roles [23]

multi-tenancy mobile network and their relationships are depicted. Network Infrastructure Provides are the owners of the physical network infrastructure, like antenna sites, hardware equipment for the antenna or local and central datacenters. They can be subdivided into Infrastructure Providers (InPs) (e.g., for the RAN or core network) and Datacenter or Cloud Infrastructure Providers. The Network Infrastructure Providers manage their physical infrastructure and virtual resources and offer them to the MSP. While the Network Infrastructure Providers provide physical network elements, the Cloud Infrastructure Providers offer virtual computing, storage and associated networking resources.
MSPs orchestrate and provide telecommunication services to end-users, also referred to as subscribers, and provide NSaaS to the tenants. Therefore, MSPs lease the required physical and virtual resources from one or several Network Infrastructure Provides. The NSaaS provided by an MSP are usually subject to a commercial SLA between an MSP and a tenant. Tenants use dedicated network slices to provide services or applications to end-users or use them for their own business operations. Note that, MSP can act as tenants by obtaining an NSI from another to expand their own NSaaS and communication service offers. Finally, the MNOs combines the roles of the MSP and the InP. They are owners and managing entity of the physical and virtual NFs (VNFs) and communication links.
The roles introduced above do not necessarily have to be assigned to separate business entities, instead it is possible that one organization holds several roles. For instance, a telecommunication service company might act as the InP, the MSP and as the MNO for a particular communication service.[23, 24]

The 3GPP 28.530 [5] and 28.801 [3] standards specify the functional business roles for network slicing slightly differently. Tenants and subscribers are subsumed as Communication Service Consumers, served by Communication Service Providers. The role of the 3GPP Communication Service Providers does not directly correspond to the MSPs in the 5G NORMA nomenclature, since the 3GPP role definitions differentiates between Communication Service Providers, Virtualized Infrastructure Service Providers and Data Center Service Providers. However, the role of the MNO as defined by 5G NORMA corresponds to the Network Operator role in the 3GPP definition.[3, 5, 23, 24]

2.2.4 Management of Network Slice Instances

In this section the most important aspects of network slice instance management, especially focusing on the network slice preparation, are given.

2.2.4.1 Network Slice Federation

Network slices are logical networks with an end-to-end scope. Therefore, they usually have to use VNFs, NFs and resources across several InPs from different domains. This is called resource and service federation. Figure 2.5 provides the big picture of resource and service federation to provide services in end-to-end network slices in

a multi-domain 5G mobile network architecture. It shows an example of a service federation of three network slices. Two of them are owned by verticals from different business sectors, the third one is owned by an MNO. Network slices 1 and 2 are using virtualized and non-virtualized NFs of several administrative domains managed by different InPs, e.g., access and core network as well as datacenters. Of course, cross-domain interaction between the used NFs is necessary.
Beyond sharing and mixing resource across different providers and domains, it is important to mention, that resources of an NSIs can be defined statically, partially dynamically or fully dynamically. Statically defined NSIs are fixed services based on non-virtualized or virtualized resources. In contrast, dynamic NSI resources provide on-demand resource allocation and scaling.[16, 20]

2.2.4.2 Network Slice Design

NSIs are described in a standardized way. It comprises the complete set of all required parameters. The GSMA Generic Network Slice Template [25] specifies a generic template for network slices, that serves as a blueprint for network slices. It defines all common slice attributes of a network slice description. Important attributes are, for example, the geographical area of service, the downlink throughput per network slice and per UE as well as the uplink throughput per network slice and per UE, the isolation level, the number of connections or terminals as well as the terminal density, the radio spectrum and the packet delay budget. The geographical area of service attribute specifies the country and regions where a network slice is available to be accessed by the local terminals. The coverage area of a network slice, might be described by a base station coverage model or an artificial geographical zone or grid partitioning abstracting from physical cell coverage areas. The downlink throughput per network slice attribute defines the downlink and uplink data rater of the network slice across all UEs. It specifies the guaranteed downlink throughput as well as the maximum downlink throughput. The same downlink throughput attributes exist for each single UE. The uplink throughput per network slice and per UE are defined similarly to the downlink throughput attributes. The required physical and logical isolation of a particular network slice from other network slices is described in the isolation level attribute. The physical isolation on the one hand refers to separation in hardware, processes and threads as well as isolated memory and network connections. On the other hand, the logical isolation regards virtual resource isolation, e.g., dedicated Virtual Machines (VMs), isolated NFs while underlying resource might be shared, or only service isolation between different tenants, while resources and NFs can be shared. The number of connections is an attribute that defines the number of concurrent sessions within a network slice. Instead, the number of terminals using the network slice simultaneously can be defined. Furthermore, the terminal density, i.e., the number of connected devices within a particular geographical area, is defined as a network slice attribute. Beyond that, the radio spectrum of a network slice has to be specified. The so-called packet delay budget network slice attribute gives an upper bound for the delay of a packet between the UE and the User Plane Function. Furthermore, related parameters like

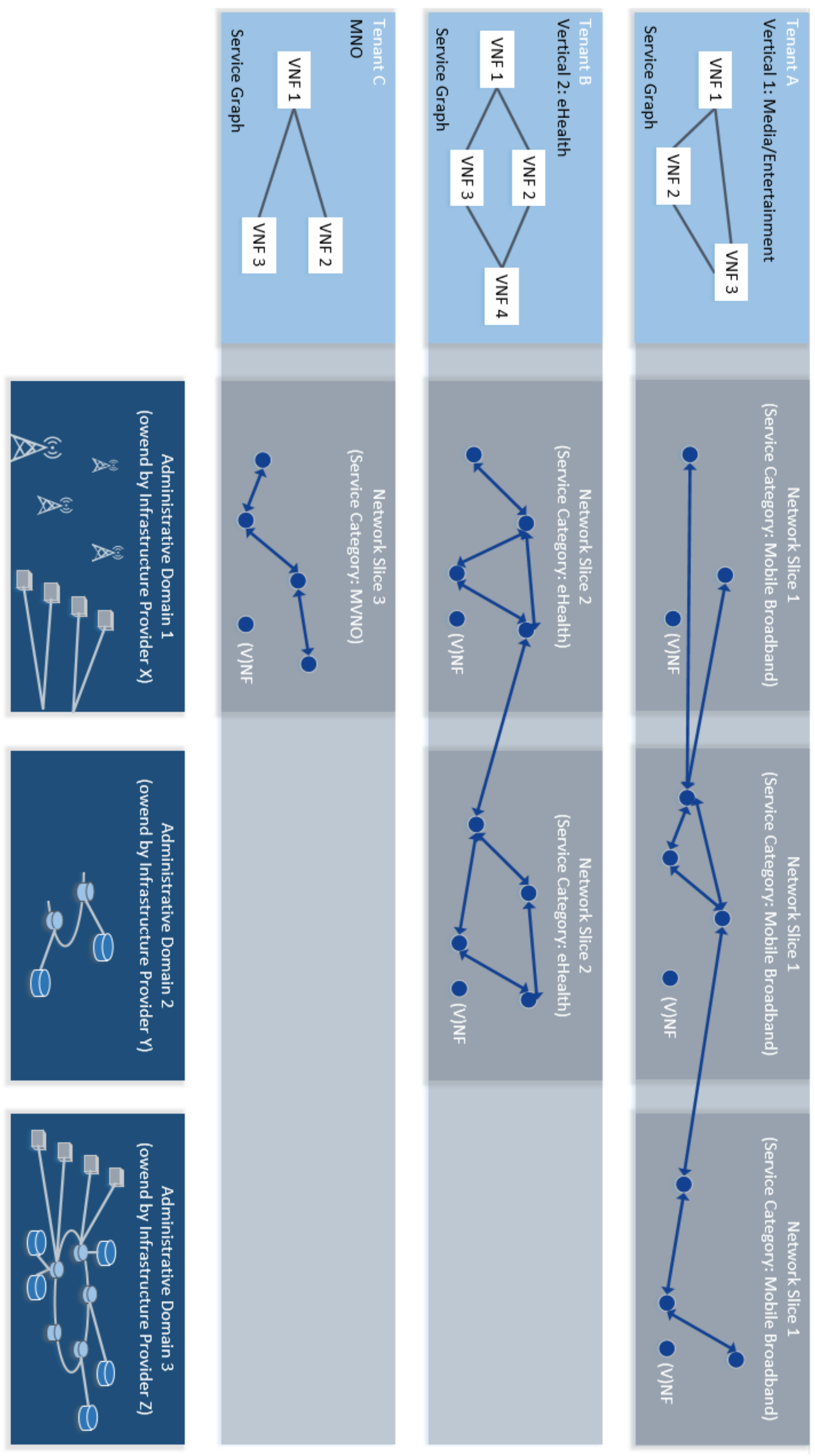


Figure 2.5: NGMN Inter-Domain Resource Integration [16]

mission critical support and slice quality of service parameters, including the packet error rate, jitter and the maximum packet loss rate, have already been specified by the GSMA.[25, 26]

An instance of a Generic Network Slice Template (GST) with concrete attribute values is called a Network Slice Type (NEST). The NEST is a network slice description using the GST attributes and filling it with concrete values. NESTs are used to define network slice requests and agree on SLA for an NSI. However, the NEST is still independent of the network design and the network slice implementation and deployment. Figure 2.6 gives an overview over the design and further usage of a

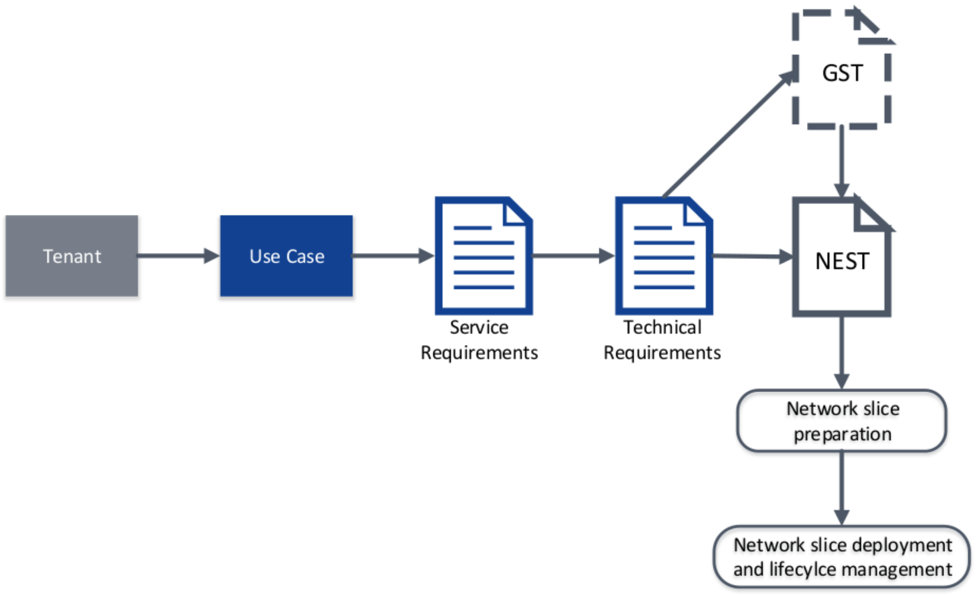

GST
Tenant
Use Case
Service Requirements
Technical Requirements
NEST
Network slice preparation
Network slice deployment and lifecylce management

Figure 2.6: NEST Definition [26]

NEST. Based on the use case, the service and technical requirements are defined and represented in the standardized NEST description. From the NEST, the MNO can derive the required resources and functions for the NSI. NESTs are the basis to decide upon the feasibility of the NSI requirements in the existing network as well as for the design of the NSI. Therefore, the GSMA NEST is fed into the 3GPP network slice preparation phase of the network slice lifecycle (see Section 2.2.4.3).[25, 26]

2.2.4.3 3GPP Network Slice Lifecycle

The 3GPP 28.530 Standard [5] specifies the NSI Management. The NSI Management spans the whole lifecycle of an NSI, comprising four phases: Preparation, Commissioning, Operation and Decommissioning. Figure 2.7 gives an overview over the management phases of an NSI and shows the high-level tasks required in each phase.

Before the actual NSI lifecycle starts, the NSI needs to be prepared. In this phase all preparatory measures required before creating and deploying a new NSI in the net-

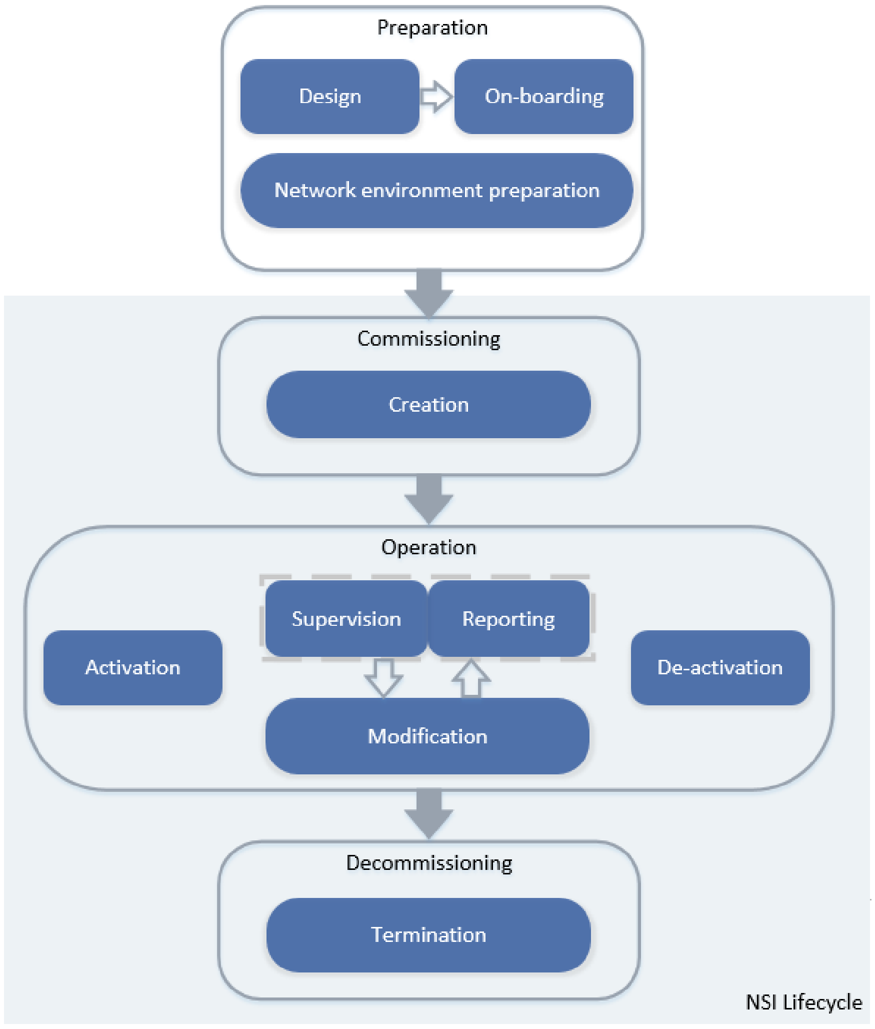


Figure 2.7: NSI Lifecycle [5]

work are carried out. This involves the design of the network slice, its on-boarding and the necessary network environment preparation. The network slice design using the GSMA GST is explained in more details in Section 2.2.4.2. Beyond that, the network slice preparation phase includes the network slice capacity planning, which is one of the major problems this thesis is contributing to.
The effective NSI lifecycle starts when the NSI is created in the so-called commissioning phase. This includes resource allocation and NSI configuration with respect to the network slice requirements. The operation phase of the NSI lifecycle starts with the activation of the created NSI. The running NSI is supervised and its performance is reported to the network slice customer. During runtime the operator must keep track of resource utilization of the NSI and monitor the NSI's performance. Manual or automated NSI modifications might be induced as a result of performance evaluation of the NSI resource usage or incoming new NSI requests. NSI modification may concern, e.g., capacity as well as topological adjustments. At the end of the NSI operation it is deactivated, i.e., its communication services are stopped. The decommissioning of an NSI involves decommissioning of its non-shared constituents and potentially undoing its NSI-specific configurations on shared constituents. After termination, the NSI does not exist anymore. [5]

2.3 Virtual Network Embedding

This section covers the foundations of the general VNE problem as well as the exact VNE algorithms used as a basis for solving the NSE problem in this thesis.

2.3.1 Introduction

Network virtualization and network slicing rely on algorithms capable of virtual network resource allocation and function placement on a substrate network provided by multiple InPs. These algorithms are commonly referred to as VNE algorithms. They solve the resource allocation problem in virtualized networks with respect to different objectives, for instance, Quality of Service (QoS) or Quality of Experience (QoE) metrics, economical profit of the InPs, security or energy-efficiency. By using VNE, existing mobile network infrastructure can be used flexibly and beneficially in virtualized mobile networks with dynamically changing virtual network topologies and fluctuating resource demands. Furthermore, automated and dynamic self-configuration and re-configuration of virtual mobile networks is crucial to enable end-to-end performance guarantees to MSPs as well as end-users. This requires highly automated resource allocation using reliable runtime and resource efficient VNE algorithms.[27]

The VNE problem can be split into two sub-problems: the virtual node mapping and the virtual link mapping problem. Virtual node mapping refers to allocating virtual nodes on physical nodes providing the demanded resources and capabilities, whereas virtual link mapping is assigning suitable physical paths to the virtual links. The virtual link mapping requires that the resources, for instance, the required throughput is available on the physical path, that capability requirements, like latency demands, are fulfilled and the sources and sinks of the virtual link and physical path match the virtual to physical node mapping. Figure 2.8 provides a simple example of a substrate composed of six connected nodes and two virtual networks. Every virtual node should be allocated on a physical node, plus each virtual link must be served by a physical path consisting of one or several sequential physical links. Virtual nodes can be allocated on any substrate node and substrate nodes can host several virtual nodes. Several virtual links can use the same physical connection. If path splitting is allowed, several suitable substrate paths can be used to serve the same virtual link. This way, the resource provisioning can be divided among several physical links.[27]

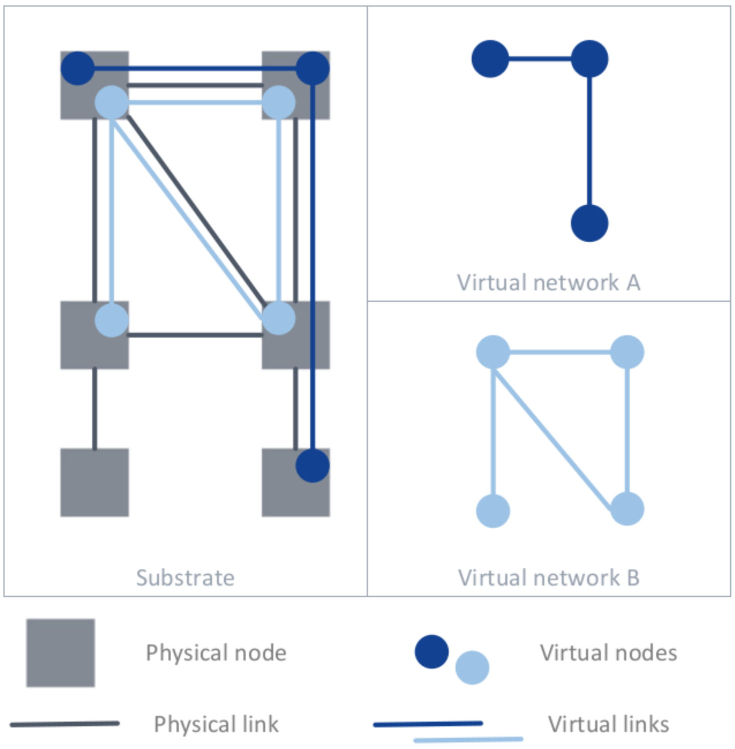


Figure 2.8: Virtual Network Embedding Overview

Due to the node and link mapping interdependency, the VNE is computationally very complex. It even belongs to the class of $\mathcal{NP}$-hard problems. In his often cited unpublished manuscript [28], Andersen shows that the VNE problem is related to the $\mathcal{NP}$-hard multi-way separator problem. However, even if a feasible node mapping is provided, the virtual link to single path allocation problem is still $\mathcal{NP}$-hard, since it can be reduced to the unsplittable flow problem (see [29]).[27]
Rost et al. provide a systematic analysis of the hardness of the VNE problem and its variants in [30]. They prove, $\mathcal{NP}$-completeness of the general VNE problem under any objective. The relaxed problem, allowing the violation of some constraints, e.g.,

capacity constraints, is shown to still be $\mathcal{NP}$-complete.[30]
Consequently, an exhaustive search for the optimal solution of a VNE problem cannot be done in polynomial time and is intractable for large problem instances [31]. Thus, heuristic and meta-heuristic approaches make up a large share of recent research in VNE algorithms [27].

2.3.2 Typical Parameters

A feasible mapping of a virtual network on a substrate network must fulfill the resource and capability requirements of the virtual nodes and links. The resources and capabilities addressed in the VNE algorithms are described as parameters of the VNE problem. The substrate elements provide individual resource capacities and have certain capabilities, while the virtual network elements have predefined capability and resource requirements. It can be distinguished between consumable and non-consumable resources. The first category is referred to as resources in the following, while the latter is referred to as capabilities. For instance, a link in the substrate network provides a certain bandwidth, while a virtual communication link requires a predefined amount of bandwidth resources. In contrast, capabilities are non-consumable characteristics of a network element, for instance, latency, level of isolation or reliability. In [32] Fischer et al. present an overview over the most important VNE parameters and in [33] Stezenbach et al. provide an extensive list of potential parameters in different categories. The most important link parameters are the maximum throughput capacity and the communication latency. In literature, often the bandwidth is used instead of the maximum throughput. Bandwidth is defined as the theoretical capacity of a communication channel while the maximum throughput describes the actual maximum data transmission capacity. Latency is sometimes referred to as propagation delay. It describes the time a data packet needs to be transferred on a specific communication link. Further relevant physical parameters of a link in a network are, for instance, the bit error rate, the technology as well as its geographical location. In addition, reliability and availability parameters, like the packet loss probability, the mean time between failure and the mean time to repair are relevant link parameters. A network node is characterized by its most important parameters: computation and memory capacities. The computation capacity, often abbreviated as Central Processing Unit (CPU) capacity, is used for routing and multiple purpose computations. In addition, the Random Access Memory (RAM) and hard disk memory capacities are highly relevant parameters for the VNE. Other relevant parameters are, for instance, the forwarding delay and capacity as well as the processing delay and the geographical location.[27, 32, 33]

Moreover, it is distinguished between so-called primary and secondary parameters. Primary parameters are resources and capabilities that can directly be specified in a virtual network request and are associated with a specific physical or virtual network element or entity. Secondary parameters are resources and capabilities depending on primary parameters or on the concrete deployment of a virtual network. Thus, secondary parameters cannot be specified in advance for a virtual network request.

For example, if a virtual link is mapped on a communication path consisting of several links in the substrate network, this might induce additional computation capacities on the passed substrate nodes. Since the selected path is not known in advance, the additional computation capacities on the intermediary substrate nodes cannot be directly specified in the virtual network request.[27, 32]
Beyond that, network parameters influenced by the network topology, like jitter (the variance in packet delay), path bandwidth or delay under changing load, are parameters in some VNE problems [33].

2.3.3 Taxonomy

VNE algorithms can be categorized, as proposed by Fischer et al. in [27]. Exact as well as heuristic solutions for solving the online or the offline version of the VNE problem exist. Online algorithms can be further differentiated between static and dynamic solutions. Beyond that, coordinated as well as uncoordinated algorithms exist and the VNE problem can be solved centralized or decentralized. The provided solution can be redundant or concise. Most of these categories are mutually independent. For example, if a VNE algorithm is online, it can still be either centralized or decentralized.[27]

Exact/Heuristic Exact VNE algorithms target at determining the optimal embedding with respect to a specific objective function, while heuristic algorithms aim at providing a good, non-necessarily optimal solution, within a short solving time.
For finding the optimal VNE, linear programming can be used. In Section 2.3.5 a performance oriented ILP-based VNE approach is provided. However, since the VNE problem is $\mathcal{NP}$-hard, see Section 2.3.1, exact algorithms do not scale for large problem instances. Large problem instances can only be solved quickly by using heuristic VNE algorithms. Pure heuristic solutions, however, usually suffer from converging to local optima. Therefore, so-called metaheuristics are frequently used. Metaheuristics are often able to overcome local optima and improve the accuracy of the solution. Examples for VNE metaheuristics are approaches based on particle swarm optimization (see, for instance, Zhang et al. [34], Guo et al. [35] and Ashraf [36]) or Max-Min Ant Colony (see, e.g., Faijari et al. [37]).[27]

Online/Offline The VNE problem often occurs in form of an online problem in practice. That means, virtual network requests usually arrive over time, i.e., future virtual network requests are not known in advance. Moreover, virtual networks can be active for an arbitrary time frame. Dynamic, online VNE algorithms are capable of reconfiguring the resource allocation and embedding of operational virtual networks, aiming at optimizing the resource utilization in the substrate network. Virtual network reconfiguration might be required due to, for instance, new virtual network requests as well as changing resource requirements, changing substrate network resource provisioning or resource fragmentation.
In contrast, offline VNE requires complete knowledge of all virtual network requests and handles all requests at once. Although offline approaches can be operated in

an online environment, offline algorithms are static. As such, they do not provide means for reconfiguration of already deployed virtual networks.[27]

Coordinated/Uncoordinated As already mentioned before, the VNE problem can be split into two main subproblems, the virtual node and the virtual link mapping subproblem. Uncoordinated VNE algorithms solve the virtual node mapping problem without considering the required communication links between the mapped nodes in the first step. In the second step the required links are tried to be embedded, given the final node mapping. However, this approach might lead to an inefficient or even infeasible bandwidth utilization, since neighboring nodes in the virtual network might be embedded on nodes in the substrate that are far apart from each other. This results in long communication paths, consuming high bandwidth resources. Therefore, coordination between the node and link mapping is desirable, especially for complex VNE problem instances.[27]

Centralized/Decentralized Centralized VNE algorithms are executed on one centralized management node with global knowledge and complete control over the whole resource allocation and virtual network embedding. Global knowledge allows to optimize the embedding globally regarding, for example, the QoS.
In centralized VNE algorithms, the centralized VNE management node is a single point of failure. This is resolved by distributed VNE approaches. Beyond that, the decentralization can improve the scalability of the VNE algorithm. Distributed VNE approaches use several distributed nodes for calculating the virtual network embedding. Those nodes do not have global knowledge, consequently the resulting embedding is not necessarily optimal. In addition, coordination between the distributed management entities is needed.
A special case of a distributed VNE algorithm is using multiple MSPs and InPs, managing different subnets. Then the overall VNE across the different subnets is distributed and requires coordination between the different MSPs and InPs.[27]

Concise/Redundant Redundancy in virtual network allocation refers to the reservation of fallback resources for substrate element failures. This is often required for fault-sensitive virtual networks. Otherwise, if a VNE algorithm does not foresee fallback elements and resources it is called a concise VNE algorithm.
Beyond that, so-called multi-path or path-splitting VNE algorithms can split the required bandwidth of a virtual communication link among several physical links in the substrate. This can be seen as a special case of redundancy, since several links are available. However, if one physical link fails, it is not guaranteed that the amount of provided resources on the remaining links is sufficient to satisfy the resource requirement of the allocated virtual links.[27]

2.3.4 Metrics and Objectives

VNE can pursue different optimization objectives. The degree of fulfillment of these objectives can be measured with the metrics introduced in this section. The most common objectives are compliance with QoS requirements, maximizing the revenue of MSPs and InPs or maximizing the robustness of the embedding.

The QoS of an embedding can be measured by using QoS metrics, for instance, path length, stress level and utilization. The path length is defined as the average number of substrate links serving one virtual link. The stress level of a substrate entity (node or link) refers to the number of virtual elements mapped to it. A more detailed metric for the element stress level is the utilization of an element. It is defined as the percentage of occupied resources of a specific resource type on a substrate element.

Maximizing the revenue of the MSPs and InPs is a natural objective for VNE. Usually maximizing the long-term revenue is perused. This objective coincides with embedding the most beneficial and as many virtual networks with a positive contribution margin as possible. Effects of the online nature of the problem, e.g., resources being blocked by less beneficial virtual networks, have to be tackled. The cost (usually defined as the sum of the prices of all used resources), the revenue (usually defined as the sum of the prices of all demanded resources by the virtual networks) and the cost-revenue ratio are obvious metrics. Moreover, the acceptance rate, i.e., the number of virtual network requests that have been accepted divided by the total number of virtual network requests within a certain time-frame, is one of the most important metrics reflecting the embedding efficiency as well as the economic profitability of a VNE algorithm.

VNE algorithms targeting at robustness or resilience of the mapping are of particular importance in the area of fault-sensitive, for instance, safety-critical use cases. That means, the embedding is optimized towards its ability to recover from node or link failure, by assigning fallback resources on backup nodes and links. If this objective is combined with the goal of efficient resource utilization it requires a trade-off between resilience and resource efficiency. Typical metrics are the number of fallback nodes, the path redundancy ratio and the cost of resilience.[27]

Further important metrics, regardless of the optimization target, are, for instance, the runtime of the algorithm, the number of required coordination messages and the number of active substrate nodes.
Apart from that, the VNE algorithms are compared with respect to the time they need to handle a virtual network request, i.e., the time needed to calculate the embedding or to determine that an embedding is currently infeasible. Runtime is a crucial factor in scenarios with dynamic virtual networks. However, quick VNE algorithms usually come at the cost of lower accuracy, which leads to lower acceptance rates and therefore lower revenues.
In distributed approaches, the coordination overhead, measured by the number of coordination messages, is an important criterion for selecting a suitable VNE algorithm.
Finally, the number of active substrate nodes is an important metric regarding the

energy-efficiency of an embedding, since unused mobile network elements can be switched off.[27]

2.3.5 Base Algorithms

A list of VNE algorithms published until 2013 can be found in the survey of Fischer et al. [27]. Newer algorithms and heuristics on the VNE problem can be found in the surveys of Cao et al., see [38] and [39].
The NSE algorithms and heuristics developed in this thesis are based on the VNE algorithm briefly introduced in the following.

The VNetMapper is an exact algorithm for solving the VNE problem using ILP, proposed by Despotovic et al. in [31]. Although the resulting ILPs are $\mathcal{NP}$-hard in general, smaller problem instances can be solved optimally with exact algorithms using, for instance, branch-and-bound, branch-and-cut or branch-and-price methods. (See [40] for more details.) However, they are not scalable for large problem instances with a large number of virtual and physical nodes and links to be mapped. The proposed approach creates an efficient solution for the VNE problem by using only binary variables and a simple objective function. The evaluation of the VNetMapper shows, that VNE problem instances with hundreds of nodes and thousands of links can be solved nearly optimally within only a few seconds. Therefore, the ILP formulation of the VNetMapper is used as a guideline for creating an ILP model for the NSE problem that allows to compute a nearly optimal solution as efficiently and scalable as possible. (See Chapter 4.)

The VNetMapper is formalized as follows: The physical substrate network is modeled as a directed graph $G_N = (\mathcal{V}_N, \mathcal{E}_N)$ with $n_N = |\mathcal{V}_N|$ physical nodes and $e_N = |\mathcal{E}_N|$ physical links. $N-1$ virtual networks should be embedded into the substrate. They are represented as directed graphs $G_k = (\mathcal{V}_k, \mathcal{E}_k)$ for $k = 1, \dots, N-1$ with $n_k = |\mathcal{V}_k|$ the number of nodes and $e_k = |\mathcal{E}_k|$ the number of links in the k-th virtual network. v_i^k and e_i^k are defined as the i-th node/link of the k-th virtual network, while v_i^N and e_i^N denote the nodes and links of the physical network. Furthermore, A_i^k for $k = 1, \dots, N$ and $i = 1, \dots, n_k$ is defined as the vectors of resource demands required by the nodes in the virtual networks (for $k < N$) and the available node resources in the substrate for $k = N$. The available and required link resources are defined similarly as the vectors E_i^k for $k = 1, \dots, N$ and $i = 1, \dots, e_k$. Thus, A_{ir}^k denotes the required/offered amount of the r-the node resource on v_i^k, while E_{jm}^k specifies the amount of required/offered m-th link resource on e_j^k. Beyond that, the VNetMapper uses the following three types of boolean variables.

$$x_{ij}^k := \begin{cases} 1 & \text{if } v_i^k \text{ is mapped on } v_j^N \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij}^k := \begin{cases} 1 & \text{if } e_i^k \text{ is mapped on } e_j^N \\ 0 & \text{otherwise} \end{cases}$$

$$y_k := \begin{cases} 1 & \text{if } G_k \text{ is embedded} \\ 0 & \text{otherwise} \end{cases}$$

It maximizes the objective function

$$f(\mathbf{y}) = \sum_{k=1}^{N-1} w_k \cdot y_k \tag{2.1}$$

with w_k the weights associated with the virtual networks, under the following constraints

$$\sum_{k=1}^{N-1} \sum_{i=1}^{n_k} x_{ij}^k A_{ir}^k \leq A_{jr}^N \qquad \forall j, r \tag{2.2}$$

$$\sum_{k=1}^{N-1} \sum_{i=1}^{e_k} z_{ij}^k E_{im}^k \leq E_{jm}^N \qquad \forall j, m \tag{2.3}$$

$$\sum_{j=1}^{n_N} x_{ij}^k = y_k \qquad \forall k, i \tag{2.4}$$

$$\sum_{i=1}^{n_k} x_{ij}^k \leq 1 \qquad \forall k, j \tag{2.5}$$

$$\begin{gathered} x_{jp} - x_{ip} = \sum_{t \in p_{in}} z_{lt} - \sum_{t \in p_{out}} z_{lt} \qquad \forall l, k, p \\ \sum_{t \in p_{out}} z_{lt}^k \leq 1 \text{ and } \sum_{t \in p_{in}} z_{lt} \leq 1 \qquad \forall l, k, p \end{gathered} \tag{2.6}$$

with p_{in} and p_{out} defined as the sets of the indices of the incoming/outgoing links of a physical node $p \in \mathcal{V}_N$.[31]

The objective function in Equation 2.1 aims at maximizing the sum of weights w_k associated with the virtual networks.
The Equations 2.2 and 2.3 specify the node and link resource capacity constraints. The map-once constraint in Equation 2.4 makes sure that every virtual node is mapped on exactly one physical node. Mapping several virtual nodes of the same virtual network on the same substrate node is ruled out by the so-called not-many-to-one constraints in Equation 2.5. This is a strong restriction which is dropped for the mobile NSE solution provided in this thesis. Finally, the graph constraints stated in Equation 2.6 assure that the structure of the virtual network with its interconnected nodes is maintained, i.e., the incoming and outgoing virtual links of a virtual node are mapped on suitable paths in the physical networks. A physical path is suitable for a virtual link if it origins from the physical node the source node of the virtual link is mapped to and ends in the physical node the sink of the virtual link is mapped to.

Even though the ILP formalization is optimized regarding runtime efficiency by using a simple objective function, boolean variables only and a set of simple constraints, the algorithm does not scale for embedding a large number of virtual networks simultaneously. Hence, only small subgroups of requests should be handled at once in practical applications.[31]

Standardized LPs, like the formal model of the VNetMapper [31], can be solved with out-of-the-box optimization software, often simply referred to as solvers. Depending on the problem instance, the chosen solver and its configuration, the optimal solution or a solution close to optimality is obtained. LP solvers usually use simplex or branch-and-cut based methods which allow them to solve even $\mathcal{NP}$-hard problems in reasonable time. Some of them are capable of parallel and/or distributed computing, offering an enormous potential for runtime improvement in multi-core and distributed computing environments. Proprietary as well as open source solvers exist. Examples for proprietary solvers are, IBM's famous CPLEX Optimizer [41], FICO's Xpress Optimization [42] and Gurobi Optimization [43]. In this thesis, the open source optimizers GNU Linear Programming Kit (GLPK) [44] and the Solving Constraint Integer Programs (SCIP) Optimization Suite [45] are used. The most commonly used solvers are briefly introduced in the following.

IBM ILOG CPLEX Optimization Studio The IBM ILOG CPLEX Optimization Studio, usually only referred to as CPLEX, has been designed to solve different mathematical optimization problems, in particular LPs, MILPs and Quadratic Programs (QPs). CPLEX uses built-in parallel computing utilizing all available cores of a computer to efficiently solve the optimization problem.[41]

Gurobi Optimization Despotovic et al. use the Gurobi solver [43] for their VNetMapper [31]. Gurobi is a fast proprietary solver for LPs and MILPs using parallel computing [43].
Some solvers, e.g., Gurobi and CPLEX even supports distributed optimization, that means, using several distributed machines to solve one optimization problem [46, 47].

GNU Linear Programming Kit The GLPK is a package comprising, for instance, the primal and dual simplex methods as well as the branch-and-cut method for solving large-scale LPs, Mixed Integer Programs and related problems [44]. The documentation of the GLPK is provided in the GLPK Manual [48].

SCIP Optimization Suite The SCIP Optimization Suite is specialized on solving Mixed Integer Programs (MIPs) using the branch-cut-and-price method. The solver allows detailed insight in and control of the solving process. The developers claim that the SCIP Optimization Suite is one of the fastest non-commercial MIP solvers. Further details on the SCIP Optimization Suite can be found in [49].[45]

3 Network Slice Instance Admission

In this chapter, the NSIA problem is analyzed in the context of the ETSI ZSM Reference Architecture Framework and the newly defined services are integrated into the framework. A formal process for the NSIA is defined based on prior art in the 3GPP standardization. The proposed process for NSIA allows to decide whether to accept or to reject an incoming Network Slice Instance Request (NSI-R) taking the individual business policies and risk tolerance of the mobile service provider into account. The approach considers careful network resource overbooking as a way of profitable network operation. The services and process defined in this chapter serve as a foundation for the NSE algorithms developed in the subsequent Chapters 4 and 5.

This chapter is organized as follows. First of all, the prior art on the Network Slice Feasibility Check standardized in 3GPP 28.531 [6] is provided. Secondly, an overview over the NSIA process is given. Finally, the NSIA services are defined and integrated into the ETSI ZSM Framework Reference Architecture [17].

3.1 Motivation and Objectives

The envisioned 5G Network Slice Customer Portal supporting NSaaS (see Chapter 1) requires full automation of the network slice lifecycle. Especially, the network slice preparation (including the NSIA), commissioning, operation and decommissioning should be automated.

Automating the NSIA is challenging as it involves embedding virtual NSIs on a shared mobile network infrastructure with uncertain resources availabilities, e.g., varying throughput availabilities and uncertain resource demands of the end-users of the NSIs. However, full automation of the NSIA is an essential foundation for providing a customer portal allowing tenants to easily configure and order new NSIs or reconfigure existing operational NSIs in the context of NSaaS.

The tenants of the 5G Network Slice Customer Portal should receive instant feedback, within only a few minutes, on the feasibility of their NSI-Rs accompanied by a cost estimation for the setup and operation of the requested NSI. In order to achieve this, manual gaps in the network slice and resource planning, feasibility checking and resource allocation must be closed. Therefore, the available network resources and capabilities of different MDs must be determined and predictions for future resource availability and resource demands must be derived from empirical values, e.g., using Machine Learning (ML) techniques and advanced forecasting methods, like time series forecasting or Hidden Markov Models. Based on that, a feasible as well as cost and resource efficient NSE, i.e., the deployment and resource allocation of the new NSI must be determined. In the course of allocating new NSIs, reconfiguring the embedding an allocation of existing, operational NSIs, in the sense of dynamic NSE, should be considered.

An automated NSIA process covering the steps of determining the available network resources and capabilities, defining the requirements of the NSIs and predicting their resource demands as well as embedding the NSIs into the physical network and allocating the required resources is presented in Section 3.3.
Beyond that, the relevant network management services for automated and quick NSI feasibility analysis in the context of the ETSI ZSM Framework Reference Architecture are defined in Section 3.4.

3.2 3GPP Network Slice Feasibility Analysis

In this section, prior art regarding the so-called network slice feasibility check, standardized by 3GPP, is summarized.
The 3GPP standard 28.531 [6] defines a high-level procedure for checking the feasibility of an NSI and reserving the required resources. The goal of this procedure is to check the feasibility of an NSI, that means, to determine, whether the network infrastructure can fulfill the requirements of an NSI, i.e., provide the needed resources and capabilities.
The 3GPP feasibility check is subdivided into the network slice feasibility check and the network slice subnet feasibility check. Both processes are briefly introduced below.

3.2.1 Network Slice Feasibility Check

The network slice feasibility check defined in 3GPP [6] has the objective to *"check the feasibility of provisioning a network slice instance to determine whether the network slice instance (NSI) requirements can be satisfied (e.g., in terms of resources)"* [6, p. 18]. It assumes that the NSI requirements have been derived from the NSI-R.

A sequence diagram of the 3GPP network slice feasibility procedure is shown in Figure 3.1. On a high abstraction level, the 3GPP Technical Specification 28.531 defines four entities involved in the network slice feasibility check and resource reservation procedure. These are the Network Slice Management Service (NSMS) Consumer and the NSMS Provider as well as the Network Slice Subnet Management Service (NSSMS) Provider and other MSPs.
The NSMS Consumer is, for example, an MSP providing NSaaS. It consumes network slice services, for instance, the network slice feasibility check and resource reservation service, offered by the so-called NSMS Provider. NSSMS Providers are

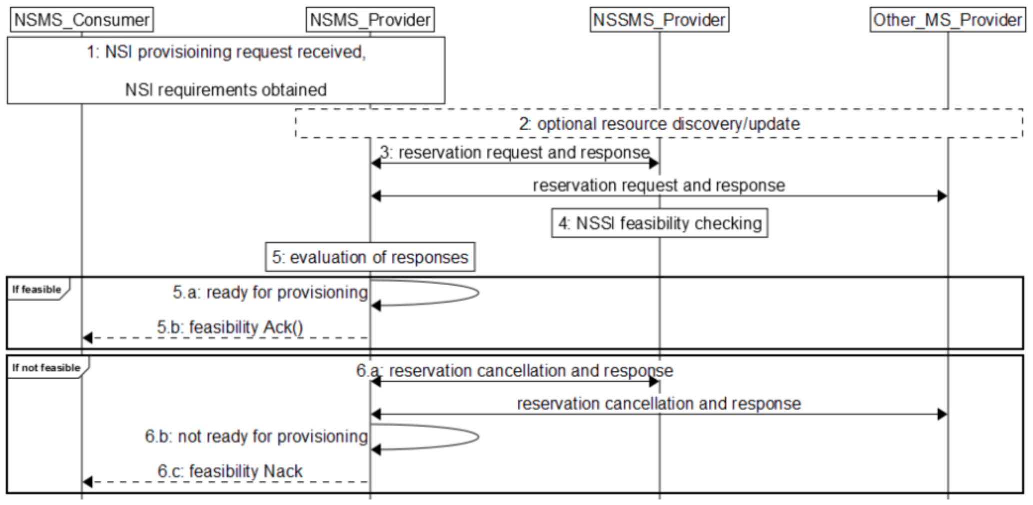

NSMS_Consumer
NSMS_Provider
NSSMS_Provider
Other_MS_Provider
1: NSI provisioining request received,
NSI requirements obtained
2: optional resource discovery/update
3: reservation request and response
reservation request and response
4: NSSI feasibility checking
5: evaluation of responses
If feasible
5.a: ready for provisioning
5.b: feasibility Ack()
If not feasible
6.a: reservation cancellation and response
reservation cancellation and response
6.b: not ready for provisioning
6.c: feasibility Nack

Figure 3.1: 3GPP NSI Feasibility Check and Resource Reservation Procedure [6]

responsible for the management services of the network slice subnet. Beyond that, 3GPP includes further MSPs.

The network slice feasibility check and resource reservation procedure comprises three major steps. Based on the NSI request the NSMS Provider identifies the required Network Slice Subnets (NSSs) and sends resource reservation requests to the according NSSMS Providers. The NSSMS Provides perform a feasibility check. In case of a positive feasibility check, they reserve the resources. Therefore, the network constituents are analyzed with respect to the required network slice resources and capabilities. Details on this so-called network slice subnet feasibility check process are provided in Section 3.2.2. The responses of the NSSMS Provides are collected and evaluated with regard to NSI feasibility by the NSMS Provider. If one or several resource reservation requests fail, the NSMS Provider might query alternative NSSMS Providers. In case the overall NSI remains infeasible, previous successful resource reservations made for this NSI must be canceled. Finally, the NSMS Consumer is notified about the feasibility of the requested NSI.[6]

3.2.2 Network Slice Subnet Feasibility Check

The feasibility check on the NSSI level is addressed in the 3GPP 28.531 standard. The NSSMS Consumer can be, for instance, an NSMS Provider requesting the resources of the NSSI when creating an NSI.
Figure 3.2 provides a sequence diagram for the NSSI feasibility check as defined in 3GPP. The NSSMS Provider receives the NSSI provisioning request including the NSSI requirements. Typical examples for NSSI requirements are, for instance, the required coverage area, the number of users and their traffic demands as well as the isolation level of the NSI. After identifying the potential MSPs, the NSSMS Provider sends resource reservation requests to the other MSPs, e.g., Management and Orchestration (MANO), Transport Network (TN) Manager, to determine the

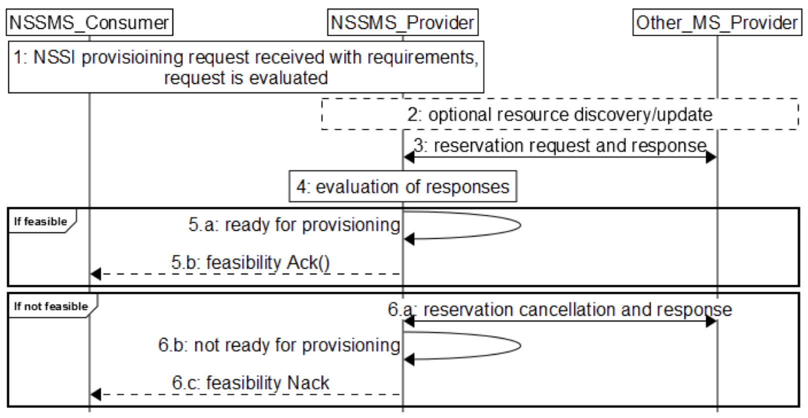

NSSMS_Consumer
NSSMS_Provider
Other_MS_Provider
1: NSSI provisioining request received with requirements, request is evaluated
2: optional resource discovery/update
3: reservation request and response
4: evaluation of responses
If feasible
5.a: ready for provisioning
5.b: feasibility Ack()
If not feasible
6.a: reservation cancellation and response
6.b: not ready for provisioning
6.c: feasibility Nack

Figure 3.2: 3GPP NSSI Feasibility Check and Resource Reservation Procedure [6]

availability of the required network constituents, for example NFs or network services. The MSPs respond with, among others, information on the availability of the requested resources. Therefore, the MSPs might evaluate the network performance information, like the load level and current as well as planned resource utilization from dedicated management data analytics services. The MSP must guarantee that existing operational services are not affected by accepting incoming resource reservations. If the NSSI requirements can be satisfied by the reserved resources the NSSMS Provider sends an acknowledgment of the successful resource reservation to the NSSMS Consumer. Otherwise, the NSSMS Consumer is informed about the infeasibility of the NSSI request and the resource reservations are canceled by the NSSMS Provider.[6]

3.3 Process for Automated Network Slice Instance Admission

The NSI and NSSI feasibility check and the resource reservation procedure, defined in 3GPP [6], are extended to a comprehensive process for the NSIA. The NSIA process is an automated decision-making process. It realizes quick decision-making on accepting or rejecting incoming NSI-Rs. As such, it is a key component of the envisioned 5G Network Slice Customer Portal. It is part of the network slice creation phase in the network slice lifecycle introduced in Section 2.2.4.3. Figure 3.3 provides a flowchart of the NSIA process on a high abstraction level.

The NSIA process starts with the NSI-R Definition step. The NSI-R Definition is done by the tenant in collaboration with the MSP. This step comprises defining the requirements of the requested NSI as well as preparing a corresponding SLA. The NSI-R requirement and SLA definition are based on a predefined GST NEST template (see Section 2.2.4.2). The GST NEST template provides the basis for a well-defined, comprehensive and standardized description of network slice requirements. For NSaaS, appropriate NEST templates for the different use cases can be

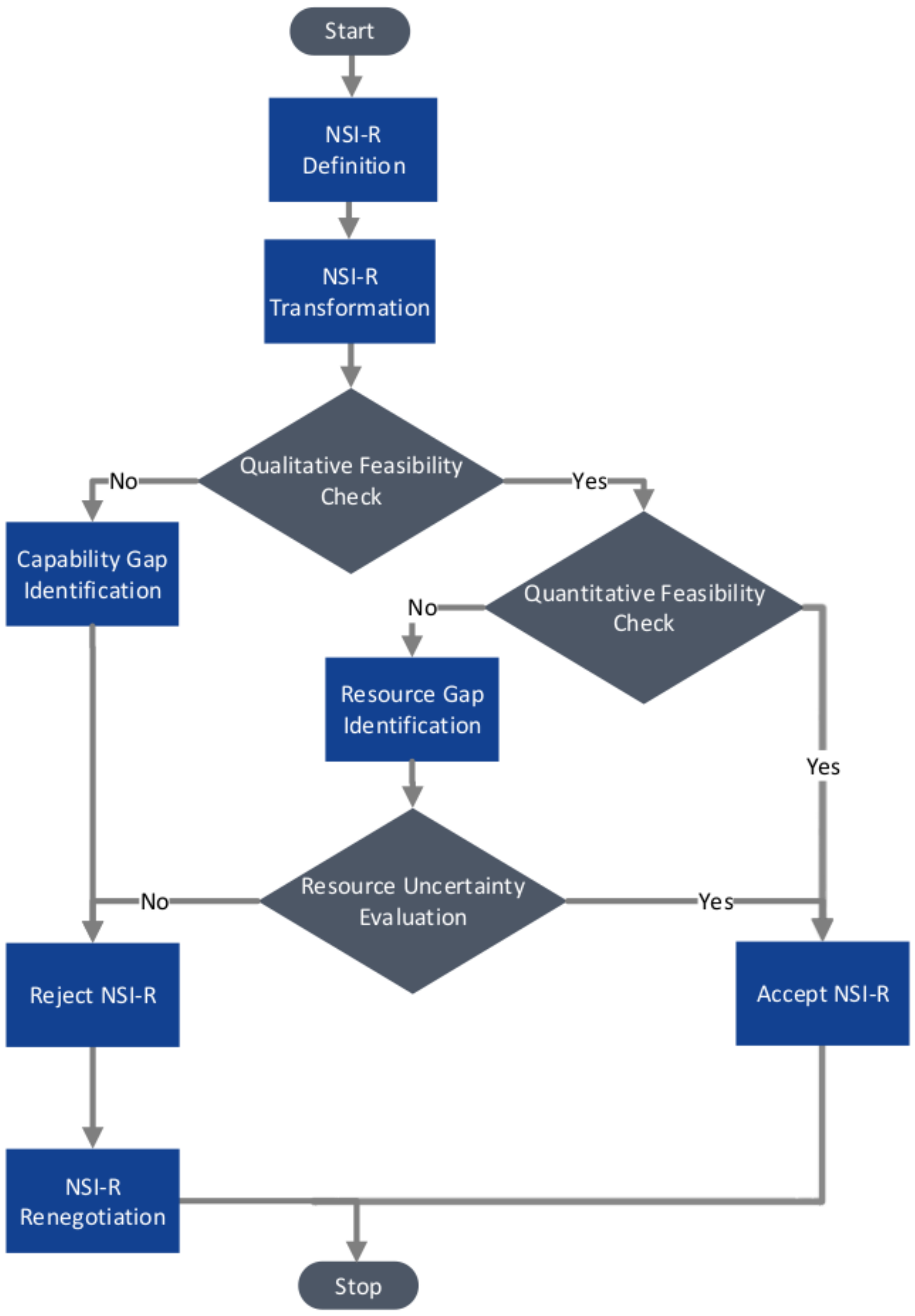

Start
NSI-R Definition
NSI-R Transformation
Qualitative Feasibility Check
No
Yes
Capability Gap Identification
Quantitative Feasibility Check
No
Yes
Resource Gap Identification
Resource Uncertainty Evaluation
No
Yes
Reject NSI-R
Accept NSI-R
NSI-R Renegotiation
Stop

Figure 3.3: Process of NSIA

used. The tenants usually define the required network slices based on one of the predefined templates. Individually designed network slices may require to be developed in close cooperation between the network slice customer and the MSP.
In the second step of the NSIA process, the so-called NSI-R Transformation, the MSP converts the NSI-R into concrete resource and capability requirements of the services involved in running the NSI-R. The result is a so-called Network Slice Instance Description (NSI-D) for the NSI-R. More details on the NSI-D are provided in Section 3.4.
These first two steps of the NSIA process, namely the NSI-R Definition and the NSI-R Transformation, are not covered by prior 3GPP Standardization in the 3GPP Network Slice Feasibility Checking procedures [6]. Nevertheless, they have to be carried out in advance of the 3GPP Network Slice Feasibility Check.

The next steps of the NSIA decision process are the Qualitative and Quantitative Feasibility Checks. They correspond with the 3GPP Network Slice and Network Slice Subnet Feasibility Check. The 3GPP feasibility checks subdivide the problem into network slices and their contained network slice subnets. However, the NSIA uses another method of division which is orthogonal to the 3GPP approach. It divides the network slice feasibility analysis into the following three sub-analysis:

- Qualitative Feasibility Check
- Quantitative Feasibility Check
- Resource Uncertainty Evaluation

This subdivision of the feasibility analysis enables a runtime-efficient, step-by-step approach.

The Qualitative Feasibility Check verifies the technical feasibility of an NSI deployment. Only the technology and network capability requirements of the NSI without considering the resources are evaluated. These technological and capability requirements regard, for example, the required RAN technology (for instance 5G), the maximum end-to-end latency or packet loss tolerance. The Qualitative Feasibility Check compares the required and provided capabilities. It can be executed very quickly without the need to consider the already running NSIs. If one or several required capabilities cannot be fulfilled by the underlying physical network, the Qualitative Feasibility Check returns a "no"-answer. If the Qualitative Feasibility Check is successful, it returns a "yes"-answer. In this case, the Quantitative Feasibility Check is executed. The preliminary Qualitative Feasibility Check of a new NSI-R allows a quick reaction to requests which are infeasible with respect to technological and network capability requirements. This improves the runtime-efficiency of the NSIA of the cases in question, since qualitative capability gaps hindering the NSI acceptance are identified quickly.

Given qualitative feasibility, the available resources in the mobile network domains as well as the required resources of the active NSIs and the estimated required resources of the new NSI-R are evaluated by the so-called Quantitative Feasibility Check. The Quantitative Feasibility Check compares the available residual resources, e.g., the

unoccupied throughput on the communication links, with the resource requirements of the NSI-D.
The Qualitative as well as the Quantitative Feasibility Checks are based on the assumption that the available mobile network infrastructure and its current and planned load can be predicted with reasonable certainty. That means, the feasibility checks of the NSIA process rely on the prerequisite that the deployed NSIs with their individual SLAs, configurations, resource demands and capability requirements as well as their actual resource utilization and capability requirements are sufficiently well known. Nevertheless, uncertainty in the resource provisioning and demand can be represented in the respective resource demand and provisioning models.

For uncertain or fluctuating resource provisioning and demands, for instance, varying resource utilization of operational NSIs in the network, the Resource Uncertainty Evaluation is executed. It identifies and quantifies the risk of SLA violation for the already running, operational as well as the new NSI-R.

If the Qualitative Feasibility Check provides a negative result, the capability gaps are identified. The shortcomings to be identified can regard, for instance, a missing service, an unavailable technology, missing coverage or a too high latency in the physical mobile network infrastructure. The identified gaps serve two different purposes. The primary intend of the capability gap identification is recognizing the causes of the NSI-R infeasibility. There might be several capabilities that would resolve the infeasibility of the NSI-R if they were enhanced or the requirements would be reduced. This knowledge, about the capability gaps can be used to renegotiate the infeasible NSI-R with the tenant and propose a mitigated, feasible version of the NSI-R and its corresponding SLA to the tenant. The second intent, of identifying capability gaps of infeasible NSI-Rs regards network planning and expansion. The identified gaps should be considered in the future network infrastructure planning and network development process of the InPs of the relevant network domains. They can serve as indicators for mid- and long-term network planning as well as mobile network infrastructure expansion. However, the identification of the capability gaps for the requested NSI-R is not always straightforward, since it often depends on the specific embedding of the virtual network slices in the physical mobile network infrastructure. Different deployments might lead to different capability gaps, for instance, one deployment might result in unsatisfiable latency constraints, while another alternative violates service availability in some areas.
If an NSI-R is qualitatively feasible, the Quantitative Feasibility Check is executed. One of the most crucial resources in end-to-end mobile networks is the throughput on the wired and wireless communication links of the physical network and on the virtual communication links of the NSIs. Throughput is defined as the actual data-rate provided by a physical communication link and utilized by a virtual communication link in the NSI. The particular importance of the throughput resource in the NSE problem derives from the fact that the throughput capacity restricts the amount of data that can be transferred between communication participants. For wired communication links the throughput capacity directly depends on the

bandwidth. However, for radio links the provided data rate on communication connections highly depends on the signal quality received by the mobile users, measured by, for instance, the Signal-to-Noise-plus-Interference Ratio (SNIR). The SNIR and thus the actual throughput are subject to, among others, UE positioning and speed as well as interferences and shadowing. The throughput utilization of an NSI depends on the data traffic of its users.

It is essential to carefully evaluate the expected available resources in the RAN, transport and core network. Past network performance, known changes in the network infrastructure, like the deployment of new antennas or BTSs can be used as a basis for the resource availability forecasting. Beyond that, the expected resource utilization of the already running, also referred to as operational, NSIs has to be analyzed in order to determine the unoccupied network resources.
The simplest variant of resource usage prediction for the incoming NSI-R is to use the amount of resources reserved for it. That means, using this estimation documented in the NSI-D, produced in the second step of the NSIA process. If empirical data on expected resource utilization is available for this particular type of NSI-R, a more precise prediction can be made.
In order to obtain better predictions, it is promising to classify NSIs and accumulate their past resource requirement data. This accumulation can be done, for instance, on an NSI use case or on a per UE basis. Accumulated and aggregated data for different NSI use cases can be used as a data basis for predicting the resource requirements of the new NSI-R or recently deployed NSIs. Moreover, predictions of the future demands of NSIs can be based on insider knowledge, for example, the information, that a particular NSI has been set up very recently and is currently in the ramp-up phase. Although its current and past resource utilization is only a small share of the booked resources it can be expected to grow considerably in the near future. These and other factors have to be taken into account to get a prediction of the remaining resources in the network that is as accurate and reliable as possible.

Once the expected resource availability on the on hand as well as the expected resource requirements on the other hand have been estimated, the virtual network slice elements must be allocated on the physical network elements. This is done by means of a VNE algorithms adapted to the particular conditions of the NSE problem. More details on the NSE algorithms are provided in Chapter 4. Of course the NSI embedding must respect the NSI capability as well as the NSI resource requirements. If the NSE algorithm can find a feasible embedding, the Quantitative Feasibility Check provides a positive response, i.e., a "yes"-answer. Otherwise, it returns a "no"-answer.

As already indicated above, available resources in end-to-end mobile networks as well as the required resources of NSIs are subject to uncertainty. Especially resources in wireless communication can be affected by fluctuations and disturbances that are hard to predict. Furthermore, radio resources, like the throughput via the so-called air-interface, are scarce resources, since frequency are limited. On that account, these resources are expensive. Thus, overprovisioning is economically inefficient. On the one hand, some use cases, for instance, URLLC, require a high network availability and reliability and network performance fluctuations might imply contracted penalty payments by the MSP or the InP and on the other hand overprovisioning of scarce resources, like RAN resources, is costly. Thus, a tradeoff between resource availability and efficient resource utilization should be perused. That means, despite possible impending penalties, an overprovisioning of RAN resource in mobile networks is economically inefficient in many cases, since mobile communication channels are very limited and the demand for mobile communication continues to grow ever more, as seen in Chapter 1. To remain profitable, overbooking of reserved, but unused network resources seems to be unavoidable, for the MSP and infrastructure provides. But such overbookings must be carefully evaluated in advance of NSI deployment, since they entail the risk of an SLA violation. The decision, whether or not a specific resource overbooking is acceptable, highly depends on the underlying business policies of the MNO.

The three most important resources for the NSIA, throughput, computation and memory capacity, are affected by the following parameters and stochastic characteristics.
The throughput provided in the RAN is subject to, among others, the available bandwidth, the used frequency bands and multiplexing technology (e.g. Time Division Multiple Access (TDMA) or Frequency Division Multiple Access (FDMA)). Beyond that, numerous environmental influences impact the signal quality and thus the provided throughput of a communication channel, for example, air temperature, foliage, rainfall, shadowing by obstacles and interference between antennas using the same frequency bands. Therefore, the provided throughput is volatile and hard to predict.
The computation power and data storage resources on the cloud servers are assumed to be constantly available, since they are provided by corresponding hardware deployed, for instance, in a data center or at the edge of the mobile networks. In contrast, the actual utilization of the computation, storage and communication resources of the deployed NSIs as well as the new NSI-R underlie variabilities due to user behavior and movement. Details on the probabilistic NSE are presented in Chapter 5.

An NSI-R might be rejected during the NSIA process due to two main reasons. The NSI-R is either qualitatively or quantitatively infeasible. In the first case, the qualitative infeasibility, the capability gaps are identified, the NSI-R in its current form is rejected. The request can be renegotiated, that means, a feasible version of the NSI-R can be offered to the tenant. This amended version of the original infeasible NSI-R has reduced requirements, which make it feasible for deployment in the network alongside with the operational NSIs. As already mentioned before, there might be several different options for fixing an infeasibility. Consequently, several different options might be presented to the customer. In the second case, the NSI-R turns out to be infeasible with regard to the quantitative feasibility check.

That means, the required resources are not available in the network. In analogy to the capability gap identification the resource gaps should be identified. If resources are short, an overbooking can be taken into consideration. In case the overbooking is acceptable, the NSI-R is accepted. Otherwise it is renegotiated or rejected. In both cases, the NSIA process ends with a decision on the NSI-R admission.

3.4 Network Slice Instance Admission Service in ETSI ZSM

The NSIA services offer the capabilities of the NSIA process introduced in Section 3.3. They address the problem of performing a time-restricted feasibility check for a requested additional NSI and constituent NSSIs. A decision on whether or not there are enough resources in the subnet domains should be made automatically within only a few minutes. Idle resources can then be used to deploy an additional NSSI or modify and reuse existing NSSIs. This decision must respect the SLAs and the QoS requirements of the operational and the additional or modified NSIs. The required short response time does not allow for querying the different potential NSSI providers for their current resource availability on demand.
In addition, the future resource availability is uncertain, especially for the RAN NSSIs. Therefore, accurate predictions on the future resource availability and QoS parameters are required to be able to decide on the feasibility of deploying additional NSIs. However, it is even more important to provide confidence values for the resource availability and the risk of SLA violation for the operational as well as the new NSIs.

The NSIA services presented in this section are based on the 3GPP standardization of the NSI Feasibility Check, see Section 3.2. The NSIA process is embedded into the multi-domain network and Service Management Reference Architecture Framework of ETSI ZSM, see Section 2.1.2.
The ETSI ZSM Reference Architecture defines an E2E Service Management Domain and several domain management areas targeting at specific network or technology domains [7]. The NSIA process comprises several services in the E2E Service Management Domain, including the main service, the so-called Network Slice Instance Feasibility Checker (NSI-FC). Figure 3.4 provides a high-level overview over the ETSI ZSM service-based network reference architecture and the integration of the NSIA services. The NSIA services, objects and roles are colored in gray, while the ETSI ZSM Reference Architecture is colored in blue.
The NSI-FC covers the three major procedures of the NSIA process, see Section 3.3. The NSI-FC comprises the Qualitative Feasibility Check, the Quantitative Feasibility Check and the Resource Uncertainty Evaluation based on an NSI-R.
While the Qualitative Feasibility Check verifies, for instance, the available technologies and network parameter configurations, the Resource Feasibility Check is responsible for evaluating the resource volume and availability in the MDs. The Resource Uncertainty Evaluation comprises a deeper stochastic risk analysis of the

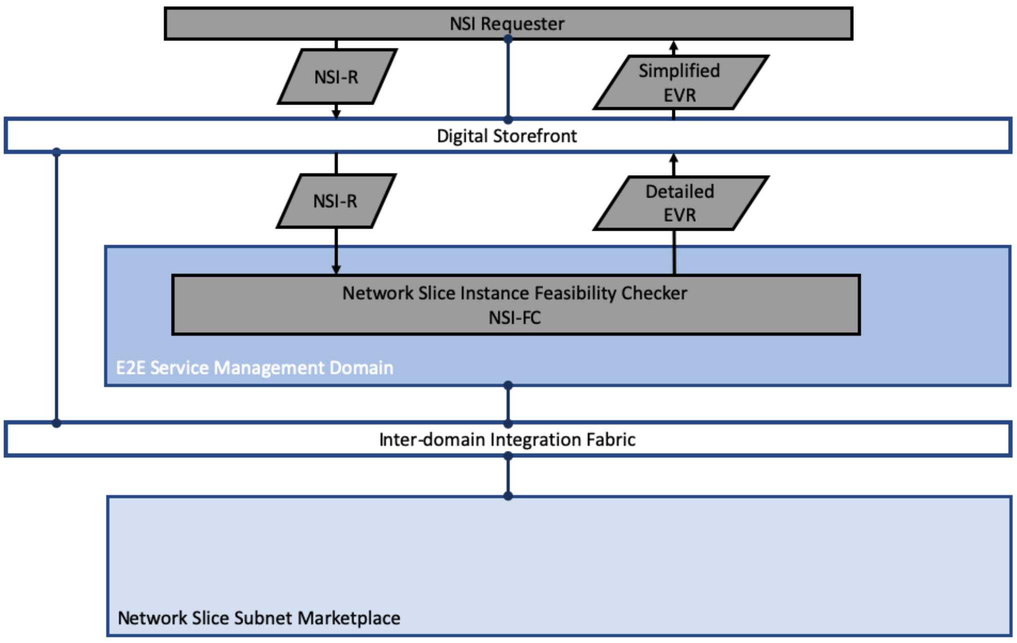

NSI Requester
NSI-R
Simplified EVR
Digital Storefront
NSI-R
Detailed EVR
Network Slice Instance Feasibility Checker NSI-FC
E2E Service Management Domain
Inter-domain Integration Fabric
Network Slice Subnet Marketplace

Figure 3.4: Components and Interfaces of NSIA

expected available and required resources. See Section 3.3 for more details.

From an architectural point of view, an NSI-R is submitted via the 5G Network Slice Customer Portal (see Figure 1.2 in Chapter 1) of the Digital Storefront, where the Network Slice Instance Provider (NSI-P) has the chance to review and potentially adapt the NSI-R in cooperation with the NSI requester, see Figure 3.4. Usually, the role of the NSI-P is carried out by the MNO.
Then the final NSI-R is passed on to the NSI-FC service that is provided by the E2E Service Management Domain. The NSI-FC takes the NSI-R as an input and derives the MD specific resources, features and configuration parameters for the required NSSI from each MD and compares them with the current resource utilization and the network performance status information provided by the respective MDs from the marketplace. Combining the information from each MD, the NSI-FC computes a detailed Evaluation Result (EVR) and returns a simplified EVR to the NSI Requester.
The Detailed EVR contains detailed information on the feasibility of deploying the requested, additional NSI in a collection of the available MDs offered in the Network Slice Subnet Marketplaces (both internal and external ones) as well as information on potential resource overbookings and confidences (in form of the probability) in the availability of the required resources and services. It contains a confidence value for being able to fulfill the resource and QoS requirements for the NSI-R. The confidence values are available at different granularity levels, for instance, on QoS parameter, resource, and network element level.
Based on the Detailed EVR, the NSI-P decides on the acceptance of the NSI-R and submits a Simplified EVR to the NSI Requester. The Simplified EVR only contains

reduced (e.g., aggregated) information about the feasibility of the NSI-R.

Figure 3.5 provides a detailed view on the NSI-FC and its integration in the ETSI ZSM Reference Architecture Framework.
Each MD exposes services for parameter, configuration and resource data provisioning, the MD Parameter & Configuration Provisioning Service and the MD Resource Data Provisioning Service towards the Intra-domain Integration Fabric. Exemplary domains include RAN, Transport and Core Network. A subset of the services can be exposed to the ETSI ZSM Inter-domain Integration Fabric. Administratively, the MDs can belong to the Network Operator (Operator's Internal Network Slice Subnet Marketplace) or to external organizations ("External/Public" Network Slice Subnet Marketplace). Each MD manages one or several NSSIs and has its own Intra-domain Integration Fabric for service registration, discovery, access control, and data exchange. Performance and resource availability data as well as the Parameters and Configuration of the MDs are separately stored in two dedicated databases for each MD. As for the other MD services, the databases can be accessed via the "Intra-domain Integration Fabric" and are partially exposed to the Inter-domain Integration Fabric.

The E2E Service Management Domain, which is, among others, responsible for the NSI Lifecycle Management, includes the NSI-FC, as depicted in Figure 3.5. The NSI-FC is called after an NSI-R has been received via the ETSI ZSM Digital Storefront. The NSI-FC manages the whole NSI embedding and feasibility check procedure by calling the other processes of the E2E Service Management Domain and requesting the required data from the services and databases in the MDs via the Inter-domain Integration Fabric. Note that, the Inter-domain Integration Fabric as well as the ETSI ZSM Digital Storefront are a set of functions and interfaces defined in the ETSI ZSM Reference Architecture.

Figure 3.6 shows the sequence diagram of the NSI request and feasibility check procedure. It explains the communication between the services.
First of all, in 1, the NSI-FC Service Consumer, in this case the NSI Requester, sends the NSI-R to the NSI-FC Provisioning Service. An NSI-R contains the following typical SLA parameters, for instance, latency, coverage, bandwidth (traffic profile), performance reliability and mobility. Additional parameters are defined in the GSMA GST (Generic Network Slicing Template) [25]. In 2, the NSI-FC Provisioning Service queries the NSI-D for the requested additional NSI from the NSI Resource and Parameter Estimation Provisioning Service, providing the NSI-R from 1. The NSI-D is a technical description of the NSI-R, based on GSMA GST. It specifies all requirements regarding resource and network capabilities. Typical Parameters are, for example, latency, throughput, computation power, memory capacities, availability and reliability. The NSI Resource and Parameter Estimation Provisioning Service responds with the NSI-D for the requested NSI to the NSI-FC Provisioning Service in 3. Then, in step 4, the NSI-FC Provisioning Service queries the Network Parameters from the Network Capability Provisioning Service. The

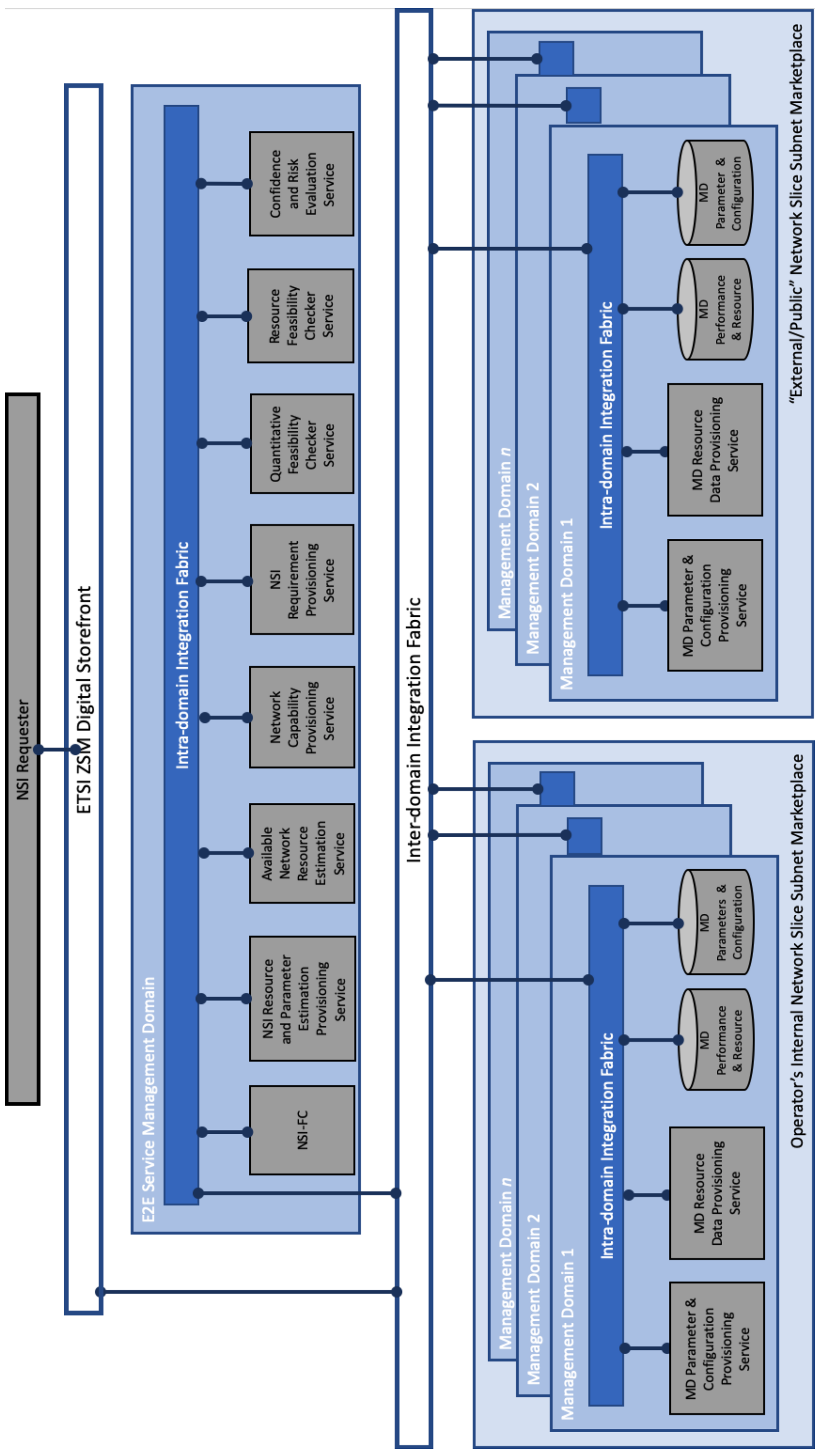

NSI Requester
ETSI ZSM Digital Storefront
E2E Service Management Domain
Intra-domain Integration Fabric
NSI-FC
NSI Resource and Parameter Estimation Provisioning Service
Available Network Resource Estimation Service
Network Capability Provisioning Service
NSI Requirement Provisioning Service
Quantitative Feasibility Checker Service
Resource Feasibility Checker Service
Confidence and Risk Evaluation Service
Inter-domain Integration Fabric
Management Domain n
Management Domain 2
Management Domain 1
Intra-domain Integration Fabric
MD Parameter & Configuration Provisioning Service
MD Resource Data Provisioning Service
MD Performance & Resource
MD Parameters & Configuration
Operator's Internal Network Slice Subnet Marketplace
Management Domain n
Management Domain 2
Management Domain 1
Intra-domain Integration Fabric
MD Parameter & Configuration Provisioning Service
MD Resource Data Provisioning Service
MD Performance & Resource
MD Parameter & Configuration
"External/Public" Network Slice Subnet Marketplace

Figure 3.5: NSI-FC and Required Services within the ETSI ZSM Architecture Framework

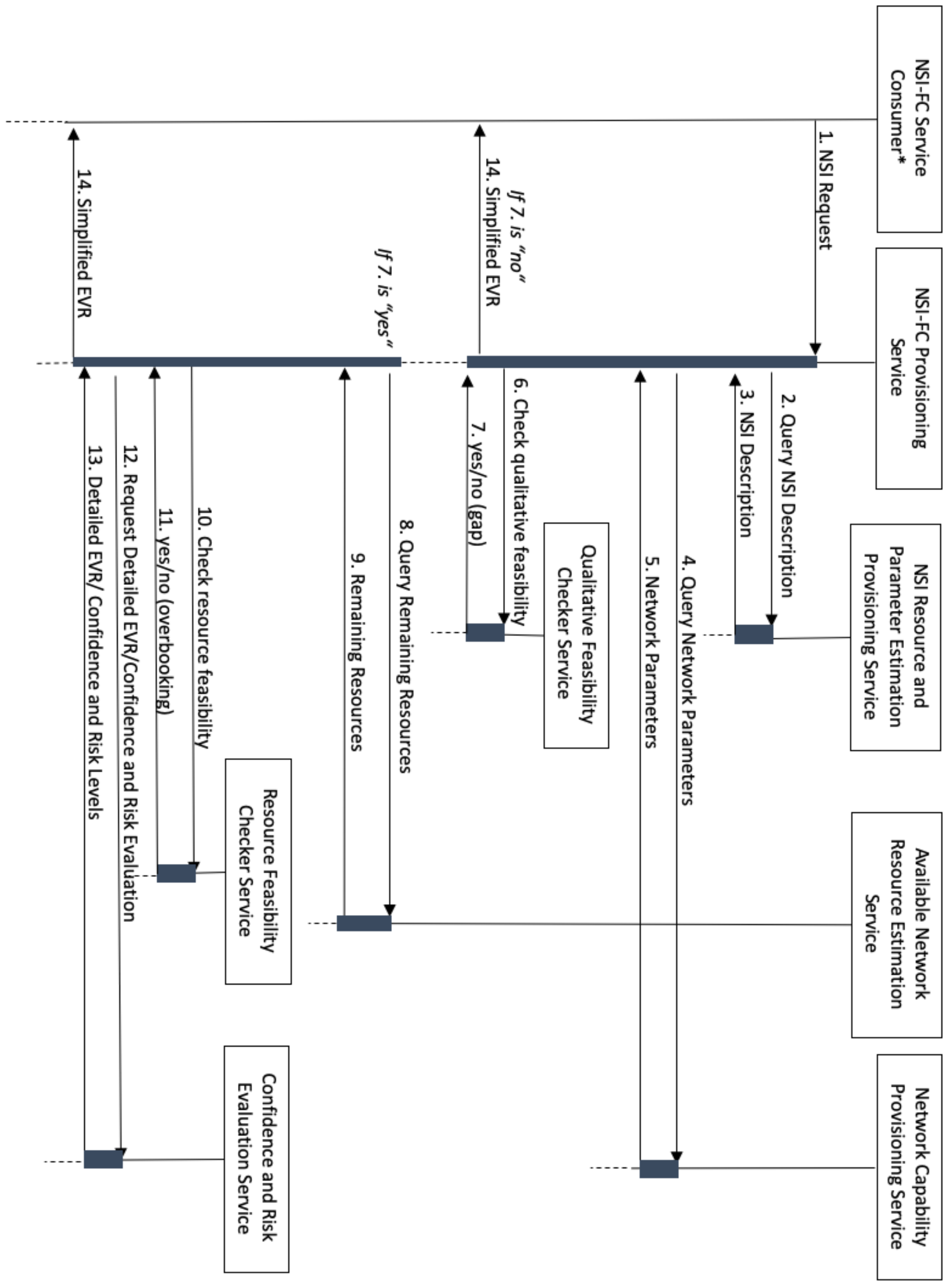

NSI-FC Service Consumer*
NSI-FC Provisioning Service
NSI Resource and Parameter Estimation Provisioning Service
Available Network Resource Estimation Service
Network Capability Provisioning Service
Qualitative Feasibility Checker Service
Resource Feasibility Checker Service
Confidence and Risk Evaluation Service
1. NSI Request
2. Query NSI Description
3. NSI Description
4. Query Network Parameters
5. Network Parameters
6. Check qualitative feasibility
7. yes/no (gap)
If 7. is "no"
14. Simplified EVR
If 7. is "yes"
8. Query Remaining Resources
9. Remaining Resources
10. Check resource feasibility
11. yes/no (overbooking)
12. Request Detailed EVR/Confidence and Risk Evaluation
13. Detailed EVR/ Confidence and Risk Levels
14. Simplified EVR

Figure 3.6: Flowchart - Execution of NSI-FC Provisioning Service

Network Parameters are selected features and configuration parameters of the Network Slice Subnet (NSS) domains, e.g., RAN technology, coverage, edge cloud availability, security features (like access control and encryption) and service and session continuity (e.g. seamless handover). In step 5, the Network Capability Provisioning Service responds with the Network Parameters to the NSI-FC Provisioning Service. Then the NSI-FC Provisioning Service triggers the Qualitative Feasibility Checker Service, providing the NSI-D as well as the Network Parameters from 3 and 5. In 7, the Qualitative Feasibility Checker Service responds to the NSI-FC Provisioning Service with “yes” or “no” and potential identified gaps to NSI requirements. If the Qualitative Feasibility Checker Service answers “no”, the process ends and the simplified EVR is returned to the NSI-FC Service Consumer, otherwise the NSI-FC Provisioning Service continues with the quantitative resource evaluation. The Simplified EVR is a “yes” or ”no” feasibility answers and other selected excerpts of the Detailed EVR. The Detailed EVR is defined as a “yes” or ”no” feasibility answer, plus the overall confidence in SLA fulfillment or the overall risk of SLA violation. For each resource, the absolute value of the expected overbooking as well as a probabilistic model for each resource is referred to as the Confidence and Risk Levels. In 8, the NSI-FC Provisioning Service queries the Remaining Resources from the Available Network Resource Estimation Service. The Available Network Resource Estimation Service reports the Remaining Resources to the NSI-FC Provisioning Service in 9. The Remaining Resources are a probabilistic model of the remaining capacity (for each consumable resource in the NSI-D) for the overall network combining the idle NSSs, when all operational NSIs are considered. Then the NSI-FC Provisioning Service triggers the Resource Feasibility Checker Service, providing the Remaining Resources from 9. In step 11, the Resource Feasibility Checker Service responds to the NSI-FC Provisioning Service with “yes” or “no” and a list of potential overbookings per resource category. The “yes” or “no” answer describes the so-called Resource Feasibility. Beyond that, the potential overbooking is provided. It is defined for each consumable resource in the NSI-D as the absolute value of the expected overbooking, as well as a probabilistic model for each resource. In the twelfth step, the NSI-FC Provisioning Service triggers the Confidence and Risk Evaluation Service, providing the Remaining Resources from 9 and the Overbooking Results from 11. Depending on its implementation, the Confidence and Risk Evaluation Service responds to the NSI-FC Provisioning Service with a Detailed EVR or only the Confidence and Risk Levels in step 13. In the last step, the NSI-FC Provisioning Service provides the Simplified EVR to the NSI-FC Service Consumer, in this case the NSI Requester.

Figure 3.7 depicts the regular queries of the subnet capabilities and NSI requirements of the already deployed NSIs.

The Available Network Resource Estimation Service queries the Network Capability Provisioning Service and the NSI Requirement Provisioning Service on a regular basis, executing the following process.
First, the Available Network Resource Estimation Service queries the Network Capacity from the Network Capability Provisioning Service. Secondly, the Network

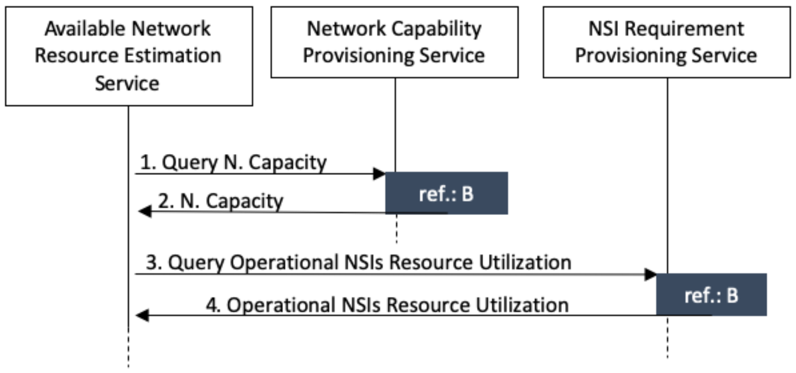

Available Network Resource Estimation Service
Network Capability Provisioning Service
NSI Requirement Provisioning Service
1. Query N. Capacity
ref.: B
2. N. Capacity
3. Query Operational NSIs Resource Utilization
4. Operational NSIs Resource Utilization
ref.: B

Figure 3.7: NSIA Sub-Process

Capability Provisioning Service provides the Network Capacity Data to the Available Network Resource Estimation Service. The Network Capacity consists of a technical description of the overall capacity of the network (combining the idle NSSs) as well as the network capabilities, features and probabilistic models of the resource availability for all consumable resources. In step 3, the Available Network Resource Estimation Service queries the Operational NSIs Resource Utilization from the NSI Requirement Provisioning Service. In the last step, the NSI Requirement Provisioning Service answers to the Available Network Resource Estimation Service with the Operational NSIs Resource Utilization. The operational NSIs Resource Utilization is defined as a probabilistic model of the resource utilization (for instance a probability distribution) for all consumable resources.

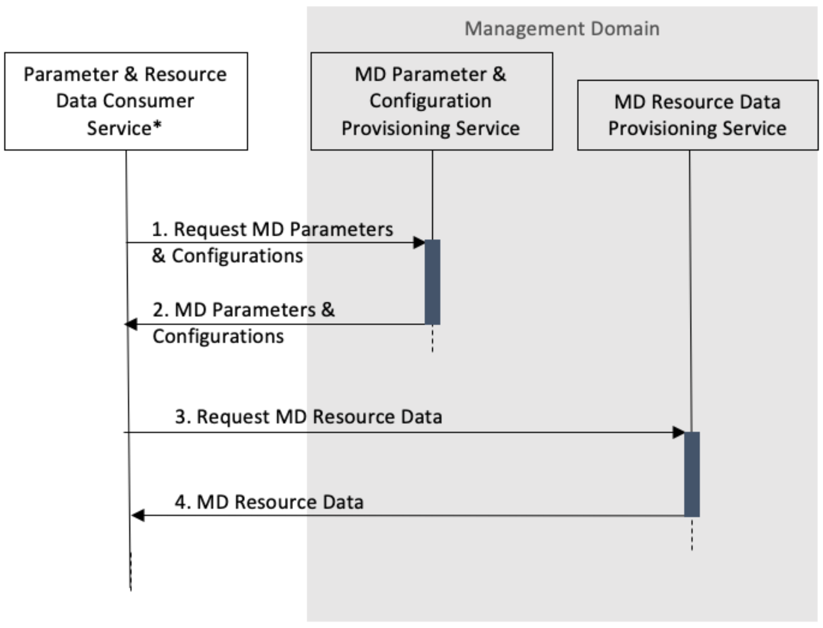

Management Domain
Parameter & Resource Data Consumer Service*
MD Parameter & Configuration Provisioning Service
MD Resource Data Provisioning Service
1. Request MD Parameters & Configurations
2. MD Parameters & Configurations
3. Request MD Resource Data
4. MD Resource Data

Figure 3.8: Sub-Process Ref. B

Figure 3.8 illustrates the sub-process, called Ref. B in Figure 3.7. The Available

Network Resource Estimation Service queries the Network Capability Provisioning Service as well as the NSI Requirement Provisioning Service on a regular basis to be able to quickly provide current network resource estimations and predictions whenever an NSI-R arrives.
Therefore, in the first step, the Parameter & Resource Data Consumer Service, in this case either the Network Capability Provisioning Service or the NSI Requirement Provisioning Service, requests the MD Parameters & Configurations from all applicable MDs. In the second step, for each queried MD, the MD Parameter & Configuration Provisioning Service responds to the Parameter & Resource Data Consumer Service with the MD Parameters & Configurations. The MD Parameter & Configurations are defined as parameters and configuration of operational NSSIs as well as the configuration possibilities of the overall MD. Typical parameters are: RAN technology, coverage, edge cloud availability, security features (like access control, encryption), service and session continuity (e.g. seamless handover), maximum number of UEs, coverage area, latency, UE mobility level and resource sharing level. In step 3, the Parameter & Resource Data Consumer Service, in this case either the Network Capability Provisioning Service or the NSI Requirement Provisioning Service, requests the MD Resource Data from all applicable MDs. Lastly, the MD Resource Data Provisioning Service responds to the Parameter & Resource Data Consumer Service with the MD Resource Data. The MD Resource Data is defined as the resource allocation and utilization of the operational NSSIs as well as the remaining capacity of the entire MD. It contains the collected data of actual resource availability for all consumable resources defined in the NSI-D for the operational NSSIs as well as idle capacity and collected data of the actual resource utilization for all consumable resources defined in the NSI-D for the operational NSSs.

4 Optimal Network Slice Embedding

This chapter introduces the models and implementation of the optimal NSE algorithm. First, the problem and motivation as well as the objectives are described. Then the formal problem definition and implementation are presented in detail and illustrated by brief, comprehensible examples.
The formalization first introduces the definitions, parameters and variables used throughout the models. Secondly, three different objective functions are presented. Finally, the three model variants of the optimal NSE are defined.
Figure 4.1 provides an overview over the variants of the NSE model. The basic NSE model serves as the foundation of the advanced path-splitting, Multiple Application Instantiation (MAI) as well as the combined NSE model.
The basic model is the simplest version, a single-path mapping approach already considering network function changing and several-to-one mappings. In the second version of the optimal NSE model path-splitting is introduced. The third variant of the optimal NSE model facilitates MEC by introducing an automated optimization for creating and allocating multiple instances of virtual network functions (referred to as applications). In the final variant both features, path-splitting and MAI, are integrated in a combined NSE model.
The objective functions specified in this chapter can be used for the model variants as visualized by the dashed lines in Figure 4.1. The simple objective function is only useful for the basic NSE model. The more sophisticated latency-aware objective function as well as the cost and revenue objective function are applicable to all model variants.

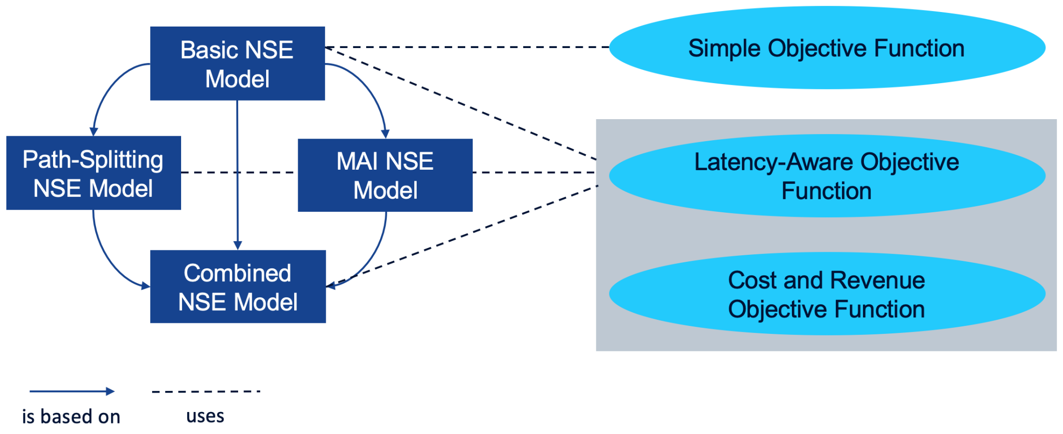

Basic NSE Model
Simple Objective Function
Path-Splitting NSE Model
MAI NSE Model
Latency-Aware Objective Function
Combined NSE Model
Cost and Revenue Objective Function
is based on
uses

Figure 4.1: Big Picture of the Optimal NSE Model Variants

4.1 Problem and Motivation

As already explained in Chapter 1, the fifth generation of mobile networks (5G) covers a wide variety of novel use cases, such as the IoT and the industry of the future (Industry 4.0), requiring mMTC. Furthermore, highly safety and security critical use cases, like autonomous driving and vehicular communication, require URLLC. But also traditional eMBB applications like HD video streaming and augmented reality must be considered. These diverse use cases introduce several radically different requirements on mobile networks. Network slicing is seen as the key concept of future 5G mobile networks, which aims at making shared networks flexible enough to cope with those divergent requirements by dissolving the traditional concept of one monolithic mobile network serving all purposes [2]. Since, network slices might be instantiated, modified or terminated dynamically or on short notice, a 5G Network Slice Customer Portal provided by a mobile service provider allowing tenants to easily configure and order new network slices is envisioned, see Chapter 1. The tenant should receive quick feedback on the feasibility of an NSI-R within only a few minutes.[3]

This chapter provides a full NSE solution for a set of NSI-Rs. It implements the quantitative and qualitative feasibility check of the NSE process, see Section 3.3. The evaluated set of NSI-Rs might include several already deployed NSIs and one or several new NSI-Rs. The NSE provides a nearly optimal allocation of the new NSI-Rs as well as a nearly optimal reallocation of the already running NSIs. If there are not enough resources to embed the new NSI-Rs, but the new ones are more profitable regarding the objective function, then the least profitable NSIs are removed in favor of the new, more profitable NSI-Rs.
If the embedding and allocation of some of the already running NSIs must remain unchanged, the occupied resource as well as the affected immutable NSIs are removed from the NSE optimization problem. This way, it is assured that the resources of the immutable NSIs remain untouched, while the already running NSIs allowing reallocation as well as the new NSI-Rs embeddings are optimized.

In order to solve the NSE problem using an LP, the physical network resources as well as the NSI resource demands are modeled as undirected graphs. The undirected graphs representing the physical network as well as the NSIs consist of nodes and edges.
For the substrate networks the nodes represent different kinds of network elements, especially groups of UEs, core-network node as well as servers and cloud computers. The UE groups are defined as sets of UEs which are connected to the same cell. The undirected edges of the substrate network graph model the communication connections between the network nodes. These can be RAN, transport links as well as core network links. Both, wired and wireless network elements are considered. However, the RAN and WiFi hot-spot resources in an end-to-end communication system is the most demanding part of the overall NSE task, since the backbone of each cell usually provides enough resources for the data transport and therefore is

usually not the primary bottleneck [50]. Network elements included in one substrate might be owned by different mobile network providers and belong to different MDs. This has the advantage that resources can be shared cross-domain and services of different network providers can be combined. This facilitates efficient utilization of all available mobile network infrastructure resources and can avoid expensive overprovisioning.
Fig. 4.2 shows a simplified end-to-end mobile network infrastructure (substrate network). The UE groups establish a radio connection to one of the radio heads or antennas. Those antennas either have their own base station or are connected to a base station via an optical or directional radio transport link (remote radio heads). Each base station can have an edge cloud. The base stations are connected to the core network by the so-called backhaul. The core network provides access to the central or main cloud. In contrast to the edge clouds, the main cloud has a large capacity for running applications and providing services.

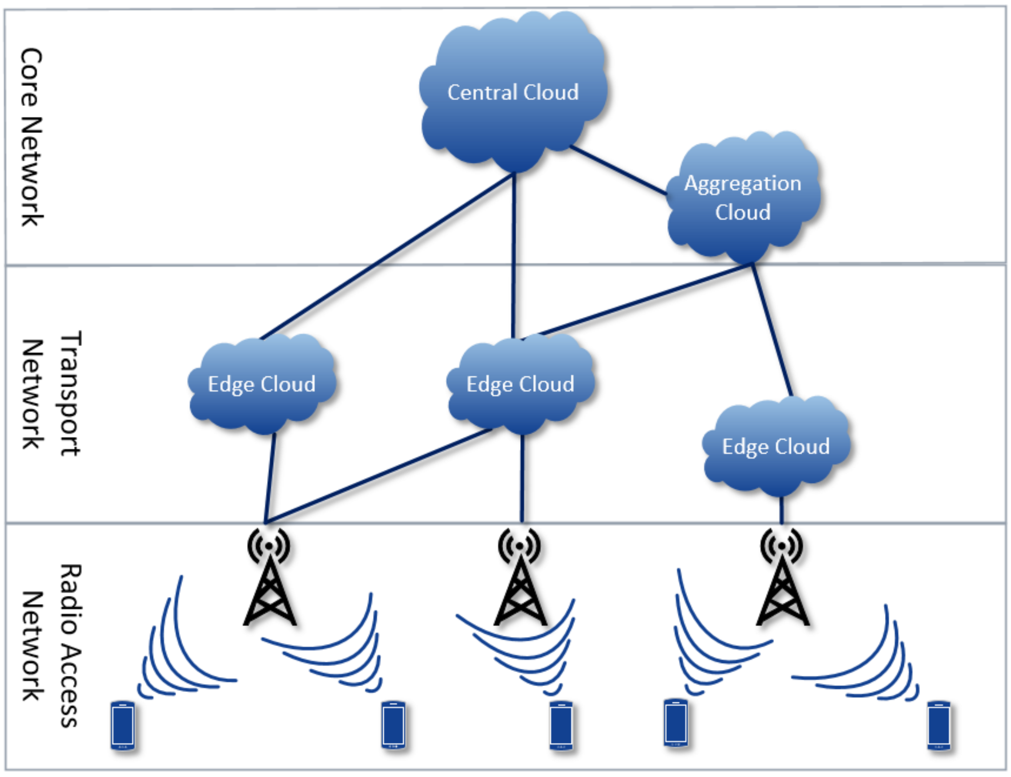


Figure 4.2: Simplified End-to-End Mobile Network Infrastructure

The graphs representing the end-to-end NSIs consist of the UE groups subscribing the services of the NSI as well as the services provided by the NSI. Those services include communication channels as well as network functions and high-level applications as defined in the NSI description. Application nodes are mobile network services like telecommunication services, establishing a connection to another end-user device or providing a service like video streaming or a virtual reality service. In comparison with the model of the physical network infrastructure the model of the NSIs is pretty simple. It only specifies the UE groups connected to the required applications. Figure 4.3 gives an example for a basic NSI with three UE groups and three applications.

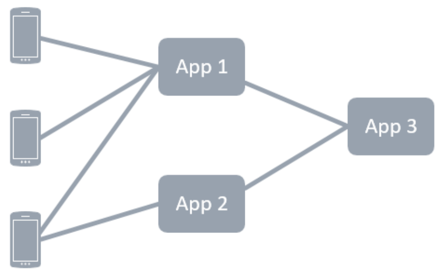


Figure 4.3: Simple End-to-End NSI

For example, in a car-to-x network slice the vehicles, the roadside-infrastructure as well as services providing, for example, information on current road and weather conditions are the application nodes of the communication network. However, modeling every single potential end-to-end connection within an NSI is not possible for most NSIs. Therefore, mobile UE connections are aggregated on cell level, i.e., several UEs connected to the same cell and using the same service are aggregated to one end-to-end connection. Obviously, the UE group nodes of the NSIs can be directly mapped onto their according representation in the substrate network. The NSI model would usually not predefine a routing for voice and data between two nodes of the virtual network in the substrate. The mapping of all links in the virtual network on the paths of the substrate has to be made by an NSE algorithm regarding the resources each link and node element of the physical network provides and the expected resource utilization of the NSIs. This also applies to so-called end-to-end connections between two UE group nodes.
Besides the mobile network and NSI topology, the provided and required resources are crucial for solving the NSE model. The most important resources and capabilities are included in the model. Nevertheless, additional resources, capabilities and restrictions can be seamlessly integrated into the model.

The most important parameters of the NSE problem are the required throughput and latency of the communication links of the physical network. But also the required CPU power and memory capacity of the cloud and mobile edge computing services. Furthermore, the reliability guarantees of the network elements are examined.
The models abstract from resource isolation and resource sharing which could be done by using, e.g., TDMA or FDMA and considers, for instance, the throughput available on a specific physical communication link as a resource to be shared among several tenants, instead.
Furthermore, it does not consider uncertainties in resource demands and resource provisioning. Throughout this chapter, it is assumed that the resources provided by the physical network as well as the resource demands of the NSIs remain constant over time. However, the signal quality of mobile communication is affected by numerous environmental influences. For instance, the available throughput resources

depend on the channel quality and the SNIR a UE experiences, which is affected by the distance between the transmitter and receiver as well as obstacles like buildings, hills or vegetation. In addition, the weather conditions as well as interferences with other antennas influence the channel quality. As a consequence, the available throughput resources vary over time in practices.
Apart from that, the actual resources utilization of the users of the network slices is time dependent. For example, the data traffic volume at daytime is very different from the data traffic volume during the night. Since the network resources can have a high variability it is helpful to annotate the graph not only with the average expected values of the resources to be allocated, but with a whole probability distribution of the resource availability. A robust solution for the NSE problem under uncertainty is presented in Chapter 5. Throughout Chapter 4, exact knowledge of the requirements, demands and capacities is assumed for all considered resource types. Beyond that, the resource requirements and resource demands are assumed to be static.
In addition, the data packet latency is simplified. The minimum time a data packet needs to be transported from the sender to the receiver is called the end-to-end latency. It depends on the used technologies and the network infrastructure hardware. The delay is the extra time a packet needs if the channels are crowded and the optimal latency cannot be achieved. The end-to-end latency plus the delay, i.e., the actual time a data packet needs to be transported from the sender to the receiver, is also referred to as latency. In this thesis a worst-case latency is assumed, which is reached when the links are on full load but not overloaded, that means, it is assumed that there is no additional delay due to congestion of the communication channels. This is a valid assumption, since overloading of the communication links is prevented by the NSE algorithm.
For simplicity, it is assumed that every application node can be deployed on an arbitrary cloud server as long as the server provides enough computation power and memory resources. In a practical context, additional restrictions might be required. The presented NSE model can handle such additional restrictions in form of dedicated constraints.

4.2 Goals and Requirements

This chapter addresses Objective 2, the development of a formalization and an efficient algorithm to solve the NSE problem under the assumption of certain resource and capability requirements and demands.

The NSE problem is a special case of the VNE problem, introduced in Section 2.3. However, NSE in end-to-end mobile networks has specific challenges. Compared to the general VNE problem, the nodes and links of the substrate network graph in the NSE problem are associated with several different and additional resource and capability parameters and other restrictions. For instance, the presented NSE models should consider network function chaining, several-to-one mappings, path-splitting

and MAI.
The vision of a 5G Network Slice Customer Portal allowing a tenant to configure new NSI-Rs and receive instant feedback on the feasibility of the requests requires a fast and efficient solution of the NSE problem. Thus, an automated and efficient NSE algorithm is needed.

This chapter aims at automating the NSIA process by modeling the NSE using an LP, which then can be solved with an out-of-the-box LP solver. The focus of the proposed model lies on resources assignment and on the allocation of network functions, user applications and services on multi-purpose cloud servers. A nearly optimal NSE on a shared physical network is determined for an arbitrary mobile network infrastructure and arbitrary NSIs.
The NSE model and algorithm presented in this chapter includes answering the following questions.

1. Are there enough resources in the physical network to embed all NSI-Rs?
2. Which NSIs should be chosen for deployment, when there are not enough resources?
3. Which cloud servers should the applications be deployed on?
4. Which communication links should be used, if there are several alternative paths?

I.e., the nearly optimal embedding answers the questions whether or not the physical network provides enough resources to serve the NSIs in its current configuration as well as which and how the NSIs should be deployed regarding the defined objectives or business goals.

4.3 Formal Problem Definition

In this section, the NSE Problem is defined mathematically. The ILP model is inspired by the VNetMapper [31], see 2.3.5. We follow the design guidelines for ILPs proposed by Despotovic et al. in [31]. The authors of [31] present a scalable algorithm for the VNE problem, the so-called VNetMapper. Their problem formulation allows to solve a VNE with hundreds of nodes and thousands of links within a few seconds. The VNetMapper is not optimized for end-to-end mobile networks. It only considers consumable resources, e.g., throughput or memory. Constraints for non-consumable resources like latency are not formalized by the approach. Beyond that, the VNetMapper does not allow to map several virtual nodes on the same physical node as well as several virtual links on the same edge of the substrate. This is not sufficient to represent an NSE, since resource sharing is one of the most important intentions of network slicing in 5G. In contrast to that, the formal model presented in this section is tailored to end-to-end mobile network slicing. It considers link latency constraints and allows several-to-one mappings. Furthermore, this approach takes advantage of the fact that the UE group nodes are the same in the physical as well as the virtual networks, i.e., the NSIs.

4.3.1 Definitions and Notation

The definitions and notation used throughout this thesis are defined in the following. A comprehensive list of the defined symbols, the List of Symbols, is attached at the end of this thesis.

4.3.1.1 General Definitions

The NSE model uses the following graph-theoretical definitions from Diestel et al. [51]. Graphs are formed by vertices and edges. This work uses undirected graphs to model the communication within physical and virtual mobile networks, since bidirectional communication is assumed on the communication links.

Definition 1 (Undirected Graph)
An undirected graph G is an ordered pair $G = (\mathcal{V}, \mathcal{E})$ comprising a set of $n \in \mathbb{N}$ vertices $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ and a set of $m \in \mathbb{N}$ edges $\mathcal{E} = \{e_1, e_2, \dots, e_m\}$.
Every edge e_k for $k = 1, \dots, m$ is associated with two ends, $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$ for $i, j = 1, \dots, n$. Edges can be denoted as $e_k := \{v_i, v_j\}$ or as $e_k := v_i v_j$.
In undirected graphs, edges do not have a direction, i.e., $v_i v_j = v_j v_i$.

A path in an undirected graph is itself a special case of an undirected graph where a set of edges forms a cycle-free sequence through all vertices of the graph.

Definition 2 (Path)
A path of length $n \in \mathbb{N}$ is an undirected graph $P = (\mathcal{V}, \mathcal{E})$ such that $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ is a set of pairwise different vertices v_i, $i = 1, \dots, n$ and $\mathcal{E}$ is a set of edges $\mathcal{E} = \{v_1 v_2, v_2 v_3, \dots v_{n-1} v_n\}$.
v_1 and v_n are called end-nodes of P.
An edge e_j is element of a path P, denoted as $e_j \in P$, if $e_j \in \mathcal{E}$.
P is well defined by its edges $P := \mathcal{E}$.

The following work frequently refers to the set of all paths between two specific vertices. For convenience, we define:

Definition 3 (Set of Paths)
$\mathcal{P}_{v_i, v_j}$ denotes the set of all paths in an undirected Graph $G = (\mathcal{V}, \mathcal{E})$ with end-nodes $v_i, v_j \in \mathcal{V}$, with $i, j = 1, \dots, n$, $n \in \mathbb{N}$ and $i \neq j$.
Note that $\mathcal{P}_{v_i, v_j} = \mathcal{P}_{v_j, v_i}$.

4.3.1.2 Mobile Network-Specific Definitions

The physical mobile network infrastructure, also referred to as the substrate in this thesis, is defined as a so-called substrate graph.

Definition 4 (Substrate Graph)
A substrate graph $N = (\mathcal{U}, \mathcal{C}, \mathcal{E})$ is defined as an undirected Graph $G = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} := \mathcal{U} \cup \mathcal{C}$. N consists of a set of UEs (User Equipments) $\mathcal{U} := \{u_1, \dots, u_n\}$ with $n \in \mathbb{N}$, a set of cloud nodes $\mathcal{C} := \{c_1, \dots, c_m\}$ with $m \in \mathbb{N}$ and a set of edges $\mathcal{E} := \{e_1, e_2, \dots, e_r\}$, $r \in \mathbb{N}$.

The vertices of the network graph are of two different types, UE groups $u_i \in \mathcal{U}$ and cloud nodes $c_v \in \mathcal{C}$ (which include edge clouds as well as central and aggregation clouds). Due to the end-to-end mobile network topology, the communication links $e_j \in \mathcal{E}$ in the network can be defined between a UE group and a cloud node or between two cloud nodes $\mathcal{E} \subseteq \{u_i c_v, c_v c_w\}$ for all $i = 1, \dots n$ and $v, w = 1, \dots, m$.

The same applies to communication paths. We define:

Definition 5 (Physical Communication Path)

A physical communication path P is defined as an undirected graph $P = (\mathcal{V}, \mathcal{E})$ in a substrate Graph N such that $\mathcal{V} = \{d, c_1, c_2, \dots, c_n\}$ with $d \in \mathcal{U} \cup \mathcal{C}$ and pairwise different nodes $c_v, c_w \in \mathcal{C}$: $d \neq c_v$ and $c_v \neq c_w$ for $v, w = 1, \dots, n$ and
$\mathcal{E} = \{dc_1, c_1c_2, c_2c_3, \dots, c_{n-1}c_n\}$.
P can be described as its consecutive edges: $P = \{dc_1, c_1c_2, c_2c_3, \dots, c_{n-1}c_n\}$.
d and c_n are referred to as the end-nodes of the physical communication path.
P is an element of the set of paths connecting d and c_n, denoted as $P \in P_{d,c_n}$ or $P \in P_{c_n,d}$.
Without loss of generality d can be referred to as the start-node, while c_n can be referred to as the end-node of P.
$\mathcal{P}$ is defined as the set of all communication paths in a network graph N.
Each communication path in the substrate network is assigned with a unique identifier $r \in \mathbb{N}$. Thus, $P_r \in \mathcal{P}$ for a unique $r \in \mathbb{N}$.

As defined above, physical communication paths are restricted to connections between a UE group and a cloud node or between two cloud nodes, but not between two UE groups. In addition, no path connects two UE groups. Therefore, at most one UE group node can be part of a physical communication path in a substrate graph.

In addition, we define the so-called free-self-links from every cloud node to itself.

Definition 6 (Free-Self-Link)

A free-self-link in a Network Graph $N = (\mathcal{U}, \mathcal{C}, \mathcal{E})$ is defined as an edge $e \in \mathcal{E}$ which forms a self-loop for a cloud node $c \in \mathcal{C}$, i.e., $e := cc$.

These links are used as paths to connect virtual applications mapped on the same physical cloud node.

Definition 7 (Free-Self-Path)

A free-self-path $P^{free}_{c_v}$ is defined as a Graph $P = (\mathcal{V}, \mathcal{E})$ in a substrate Graph N, such that $\mathcal{V} = \{c_v\}$ for a $c_v \in \mathcal{C}$ and $\mathcal{E} = \{c_v c_v\}$.
The set of free-self-paths in N is denoted as $\mathcal{P}_{free}$.

The free-self-paths are used to accommodate the communication between virtual nodes mapped on the same physical node.

NSI-Rs that should be embedded into the mobile network infrastructure are received over time via the 5G Network Slice Customer Portal. The NSI-Rs are numbered in ascending order as they arrive, with a $k = 1, \dots, n$ with $n \in \mathbb{N}$. If two or several NSI-Rs arrive concurrently, they are sorted randomly to receive a unique NSI-R number. An NSI-R is formally defined as follows.

Definition 8 (Network Slice Instance)
The k-th NSI for a $k = 1, \dots, n$ with $n \in \mathbb{N}$ is defined as an undirected Graph $G = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} := \mathcal{U}_k \cup \mathcal{A}_k$ and $\mathcal{E} = \mathcal{L}_k$.
N_k is associated with a weight $\omega_k > 0$ and defined as a network graph $N_k = (\mathcal{U}_k, \mathcal{A}_k, \mathcal{L}_k)$ for $k = 1, .., n$ with $\mathcal{U}_k \subseteq \mathcal{U}$, $\mathcal{A}_k = \{a_1^k, a_2^k, \dots, a_{m_k}^k\}$, $m_k \in \mathbb{N}$ the applications and $\mathcal{L}_k = \{l_1^k, l_2^k, \dots, l_{r_k}^k\}$, $r_k \in \mathbb{N}$ the virtual communication links.

The NSIs are modeled as virtual network graphs $N_k = (\mathcal{U}_k, \mathcal{A}_k, \mathcal{L}_k)$ for $k = 1, .., n$ with $\mathcal{U}_k \subseteq \mathcal{U}$, that means, the UE groups are already embedded in the physical network by definition.
To compare the importance of the NSIs relative to each other, for instance, the utility of the NSIs or their revenue for the InP, weights ω_k are assigned to them.
$\mathcal{A}_k$ is the set of application nodes of the k-th NSI, with $a_m^k \in \mathcal{A}_k$ defined as the m-th application node of N_k. To guarantee NSI isolation, NSIs never share application instances. Even if they require the same type of application, distinct instances of the same application are defined for the NSIs.
The virtual communication links connecting UE groups and application nodes, are defined as $l_i^k \in \mathcal{L}_k$ for the k-th NSI. In the NSI no direct UE group to UE group connections are allowed. A communication connection, like a phone call, between two UE groups is modeled as two connections, one for each UE group towards a common application. Without loss of generality, every l_i^k can be written as $l_i^k = \{u_v, a_m^k\}$ if it is a UE group to application connection or as $l_i^k = \{a_q^k, a_m^k\}$ if it is an application to application connection for distinct applications with $q \neq m$.

Virtual communication paths are the equivalent of physical communication paths in a virtual network graph or network slice. They are defined as follows:

Definition 9 (Virtual Communication Path)
A virtual communication path P^k is defined as an undirected graph $P^k = (\mathcal{V}, \mathcal{E})$ in an NSI N_k such that $\mathcal{V} = \{d, a_1, a_2, \dots, a_n\}$ with $d \in \mathcal{U}_k \cup \mathcal{A}_k$ and pairwise different nodes $a_m \in \mathcal{A}$: $d \neq a_m$ and $a_m \neq a_q$ for $m, q = 1, \dots, n$ and $\mathcal{E} = \{da_1, a_1a_2, a_2a_3, \dots, a_{n-1}a_n\}$.
d and a_n are referred to as the end-nodes of the virtual communication path.
P^k can be described as its consecutive edges: $P = \{da_1, a_1a_2, a_2a_3, \dots, a_{n-1}a_n\}$.
P^k is an element of the set of paths connecting d and a_n, denoted as $P^k \in P_{d,a_n}^k$ or $P^k \in P_{a_n,d}^k$.
Without loss of generality d can be referred to as the start-node, while a_n can be referred to as the end-node of P^k.
$\mathcal{P}$ is defined as the set of all communication paths in an NSI N_k.
Each communication path in the NSI is assigned with a unique identifier $r \in \mathbb{N}$. Thus, $P_{d,a_n}^k = P_{a_n,d}^k$ can also be referred to as P_r^k for a unique $r \in \mathbb{N}$.

4.3.2 Parameters

NSI acceptance implies a feasible NSI embedding and resource assignment. The most important resources and capabilities, identified in Section 4.1, are considered in the ILP formalization.
Throughout this thesis, we distinguish between node and link resources and capabilities. Resources are consumable capacities provided by a network element, while capabilities are non-consumable characteristics of the network elements.
The most important resources provided by the physical network nodes (edge, aggregation and central clouds) are computation power and memory. Furthermore, the node reliability is taken into account as a node capability. The virtual link mapping is, among others, restricted by throughput resources as well as the link latency and reliability.
Only the most important resources and capabilities of the nodes and links in the network are considered in this work. However, adding further resources and capabilities is straightforward, since the following model includes generalized constraints which can be used as templates for additional parameters.

4.3.2.1 Node Resources and Capabilities

The edge, aggregation and central cloud servers provide resources and capabilities which can be assigned to applications of the NSIs. A general resource of a node $c_w \in \mathcal{C}$ in the substrate network is denoted as O_w. I.e., O_w represents an arbitrary node resource. Concrete instances of provided physical node resources considered in this thesis are: The computation and memory capacity of a cloud node c_w, defined as D_w and M_w respectively. The general notation for capabilities provided by a substrate node c_w is W_w. We use the node reliability as an example for a node capability. It is denoted as B_w in the following.
Complementary to the resources and capabilities provided by the physical mobile network (substrate), the NSIs come with resource and capability requirements that should be fulfilled by the physical network elements. The required resources of an application node a_m^k of the NSI N_k is defined as O_m^k. O_m^k can represent either the required computation power D_m^k or the required memory capacity M_m^k of a_m^k. In general, the capability requirements of a node a_m^k in an NSI N_k is denoted as W_w^k. The required reliability B_m^k, defined for the hardware the application a_m^k is deployed on, serves as a concrete example for a node capability requirement.
Table 4.1 provides an overview over the previously introduced notation regarding the node parameters, used throughout this thesis.

4.3.2.2 Link Resources and Capabilities

The link resources are similarly defined. On the one hand, there are the substrate resources and capabilities, on the other hand we have the resources and capabilities demanded by the NSI-Rs. In a general notation, the provided link resources are referred to as R_j for a physical edge e_j while the required link resources are called

Table 4.1: Node Resources and Capabilities

Def.	**Type**	**Description**
O_w	General Resource	Provided resource of substrate node c_w
D_w	Specific Resource	Provided computation capacity of substrate node c_w
M_w	Specific Resource	Provided memory capacity of substrate node c_w
W_w	General Capability	Provided capability of substrate node c_w
B_w	Specific Capability	Provided reliability of substrate node c_w
O_m^k	General Resource	Required resource of node a_m^k in N_k
D_m^k	Specific Resource	Required computation capacity of node a_m^k in N_k
M_m^k	Specific Resource	Required memory capacity of node a_m^k in N_k
W_m^k	General Capability	Required capability of node a_m^k in N_k
B_m^k	Specific Capability	Required reliability of node a_m^k in N_k

R_i^k for a virtual link l_i^k which belongs to an NSI request N_k. Similarly, an arbitrary capability provided by a physical edge e_j is referred to as Q_j while its network slice counterpart is called Q_i^k for a link l_i^k.
Specific link resource capacities and demands of end-to-end mobile networks considered in this model are the provided and demanded throughput T_i for a physical communication link e_j and T_i^k for a virtual communication link l_i^k in an NSI request N_k, as well as the link latency and reliability. Table 4.2 summarizes further details on these parameters. The available throughput is defined as the maximum possible

Table 4.2: Link Resources and Capabilities

Def.	**Type**	**Description**
R_j	General Resource	Provided resource of substrate edge e_j
T_j	Specific Resource	Provided throughput of substrate edge e_j
Q_j	General Capability	Provided capability of substrate edge e_j
L_j	Specific Capability	Latency on substrate edge e_j
A_j	Specific Capability	Reliability of substrate edge e_j
R_i^k	General Resource	Required resource of link l_i^k in N_k
T_i^k	Specific Resource	Required throughput of link l_i^k in N_k
Q_i^k	General Capability	Required capability of link l_i^k in N_k
L_i^k	Specific Capability	Required latency of link l_i^k in N_k
A_i^k	Specific Capability	Required reliability of link l_i^k in N_k

throughput on the wired or wireless communication link $e_j \in \mathcal{E}$ of the network infrastructure. To keep the model simple, uplink and downlink are not distinguished.

For RAN connections an expected Channel Quality Index (CQI) has to be defined in advance in order to determine the maximum throughput based on the frequency bandwidth.
For simplicity, the maximum latency is modeled as an upper bound for the data transmission time if the link is on full load, but not overloaded.

Finally, the resources and capabilities of the free-self-links (see Definition 6) are specified. As the free-self-links represent node-internal communication between applications, the resources and capabilities of these imaginary links are unlimited by definition.

Definition 10 (Free-Self-Link Parameters)
Free-self-links provide unlimited resources and perfect capabilities.

Thus, free-self-links do not cause bottlenecks. For a free-self-link e_j in the substrate we define for our specific parameters, as introduced above: The throughput of a free-self-link e_j is unlimited: $T_j := \infty$, its latency is zero $L_j := 0$ and its reliability is 100% $A_j := 1$.
Consequently, the free-self-paths (see Definition 7) can be used without limitations and costs. However, if the internal communication on a cloud node is subject to restrictions or constraints, i.e., due to the internal bus-system, restricted instead of free links can be used by setting the parameters to the respective limits. Node-internal communication restrictions are however out of the scope of this thesis.

Table 4.3: Path Resources and Capabilities

Def.	**Type**	**Description**
R_{P_r}	General Resource	Provided resource of substrate path P_r
T_{P_r}	Specific Resource	Provided throughput of substrate path P_r
Q_{P_r}	General Capability	Provided capability of substrate path P_r
L_{P_r}	Specific Capability	Latency on substrate path P_r
A_{P_r}	Specific Capability	Reliability of substrate path P_r

The link resources and capabilities, defined in Table 4.2, are extended on paths, see Table 4.3. The aggregation of the resources and capabilities on path level depends on the characteristics of the resources and capabilities.
For example, the throughput of a path P_r with $\mathcal{E} = \{e_0, e_1, \ldots, e_m\}$ is defined as the minimum throughput of the involved edges:

$$T_{P_r} := \min_{j=0,\ldots,m} T_j$$

The aggregated latency L_{P_r} of P_r is defined as the sum of the latency of its edges:

$$L_{P_r} := \sum_{j=0}^{m} L_j$$

Finally, the reliability of a path is defined as the minimum reliability of its edges:

$$A_{P_r} := \min_{j=0,\dots,m} A_j$$

4.3.3 Variables

The NSE problem is based on three variable types. The first type of variables are the embedding variables y_k, one for each NSI.

Definition 11 (Embedding Variables)
We define $n \in \mathbb{N}$ embedding variables y_k, for $k = 1, \dots n$, one for each NSI-R as

$$y_k := \begin{cases} 1 & \textit{if } N_k \textit{ is embedded in } N \\ 0 & \textit{otherwise} \end{cases}$$

y_k is set to 1 if N_k is accepted and embedded into the substrate. Otherwise, $y_k = 0$ if the k-th NSI-R is rejected.
Thus, y_k is a binary variable.

The second and third types of variables mapping variables. It is distinguished between node and link mapping variables.

Definition 12 (Node Mapping Variables)
We define an application (NSI application node) to cloud (substrate node) mapping $a2c^k_{mw}$ variable for every substrate node and every application node in all NSI-Rs as

$$a2c^k_{m,w} := \begin{cases} 1 & \textit{if } a^k_m \textit{ is mapped on } c_w \\ 0 & \textit{otherwise} \end{cases}$$

The node mapping variable $a2c^k_{m,w} = 1$ if and only if the virtual application node a^k_m is mapped on the physical cloud node c_w. $a2c^k_{m,w}$ is also a binary variable.

Definition 13 (Binary Link Mapping Variables)
We define the virtual link (link in NSI) to communication path mappings $l2p^k_{i,r}$ for every virtual link in the NSIs and communication path in the substrate P_r as:

$$l2p^k_{i,r} := \begin{cases} 1 & \textit{if } l^k_i \textit{ is mapped on } P_r \\ 0 & \textit{otherwise} \end{cases}$$

The link mapping variables $l2p^k_{i,r}$ describe the mapping of virtual links l^k_i on physical substrate paths P_r. If the link l^k_i is mapped on the path P_r, then $l2p^k_{i,r} = 1$, otherwise, the mapping variable is set to 0.
Since the basic elements of the substrate network model are nodes and edges, we also define the physical path to edge mapping.

Definition 14 (Path to Edge Mapping)
We define the substrate communication path P_r to substrate edge mapping as:

$$p2e_{r,j} := \begin{cases} 1 & \text{if } e_j \text{ is used in } P_r \\ 0 & \text{otherwise} \end{cases}$$

This mapping is constant and can be derived directly from the substrate paths. $p2e_{r,j} = 1$ if and only if the physical path with ID r includes the physical link e_j, $p2e_{r,j} = 0$ otherwise.
Together with the $l2p$ mapping variables, the $l2e$ mapping can be derived without creating any additional variables. The $l2e$ mapping is not necessary for the model. However, it supports a better understanding of some relations and constraints used in the following.

Definition 15 (Link to Edge Mapping)
We define the NSI link to substrate edge mapping as

$$l2e^k_{i,j} := \sum_{\forall r} \left(l2p^k_{i,r} \cdot p2e_{r,j} \right)$$

The $l2e^k_{i,j}$ mapping is 1 if and only if the $l2p^k_{i,r} = 1$ and $p2e_{r,j} = 1$, that means, if l^k_i is mapped on P_r and P_r includes e_j. Otherwise, $l2e^k_{i,j}$ is 0.

In the basic version of the NSE problem formalization without using multi-path methods all variables are binary. Thus, the NSE can be modeled as an ILP, i.e., an LP with integer variables only.

4.3.4 Objective Functions

The utility of an NSE is measured with regard to an objective function. In this section, three specific objective functions are introduced.

4.3.4.1 Simple Objective Function

The basic objective function is the so-called simple objective function (see [31]).

Definition 16 (Simple Objective Function)
The simple objective function is defined as

$$\max_{y_k} \left(\sum_{\forall k} y_k \frac{\omega_k}{\phi} \right)$$

with the normalization factor

$$\phi := \sum_{\forall k} \omega_k$$

The simple objective function, as defined above, maximizes the sum of weights ω_k of the embedded NSIs. The weights ω_k are normalized by the constant ϕ, such that the weights sum up to 1.
The NSI weights ω_k allow a great degree of freedom. In the simplest version, equal weights of 1 are assigned to all NSIs. In this case, the algorithm simply targets at embedding as many NSI-Rs as possible, without prioritizing them.
The NSI weights can be used to prioritize the NSIs, for instance, the revenue of the NSI for the MSP. Higher NSI weights mean a higher priority of the associated NSI-R.

4.3.4.2 Latency-Aware Objective Function

The latency-aware objective function is an enhancement of the simple objective function introduced in Section 4.3.4.1.

Definition 17 (Latency-Aware Objective Function)
The latency-aware objective function is defined as

$$\max_{y_k, l2p^k_{i,r}} \left(\left(\sum_{\forall k} y_k \frac{\omega_k}{\min\limits_k \omega_k} \right) - \left(\frac{\sum\limits_{\forall k,i,r} l2p^k_{i,r} \cdot L_{P_r}}{\sum\limits_{\forall k,i,r} l2p'^k_{i,r} \cdot L_{P_r}} \right) \right)$$

with

$$l2p'^k_{i,r} := 1, \, \forall i, r, k$$

The latency-aware objective function comprises two terms. The first term is similar to the simple objective function (see Definition 16), but not normalized to 1. However, it is still responsible from embedding as many NSIs as feasible taking their priorities into account. The NSI priorities are modeled by the NSI weights. I.e., the first term of the latency-aware objective function is responsible for maximizing the sum of the weights of the embedded NSIs. Instead of the normalization used in Definition 16, the weights ω_k are scaled so that the smallest weight is 1 and the other weights respect the relative importance given by the original weights. For example, if the weights are $\omega_0 = 2$, $\omega_1 = 4$ and $\omega_2 = 5$ then the weights are scaled by the factor $\frac{1}{\min\limits_k \omega_k} = \frac{1}{2}$. Thus, the scaled weights are $\omega'_0 = 1, \omega'_1 = 2$ and $\omega_2 = 2.5$.
As a result, each embedded NSI adds a benefit of at least 1 to the latency-aware objective function.
This embedding objective is augmented by a second, latency-aware term. The sum of all latencies on the used substrate links is minimized. Minimization is achieved by the factor -1 multiplied to the term, since latencies and the mapping variables are positive values. In the numerator of the second term of the latency-aware objective function all latencies of the used physical links are summed up. The denominator normalizes the sum of the latency-cost factors to 1. The set of parameters $l2p'^k_{i.r} := 1$ for all i, r and k is defined for the notation of the sum only.
As a result, the utility of embedding any NSI is always at least as high (normalized

weight of 1) as all latency costs for all NSI embeddings together (sum of weights is 1).

4.3.4.3 Cost and Revenue Objective Function

Approaching the NSE problem from a business perspective, a cost and revenue optimization is often required.
NSI-R acceptance underlies technical as well as economic considerations. That means, NSI-Rs must be technically feasible, but also economically beneficial to be accepted by the MSPs and deployed in the mobile network. However, the true costs of mobile network usage are often unclear, as the operation of a network naturally entails a high proportion of fixed costs, like costs for capital, rent of BTS sites and personnel costs. Since the allocation of these costs to a specific network usage or an NSI is not straightforward, the following cost and revenue objective function, which is based on relative revenue and costs, is used in this thesis.

Definition 18 (Cost and Revenue Objective Function)
The cost and revenue objective function is defined as

$$\max_{y_k, a2c^k_{m,w}, l2p^k_{i,r}} \left(\left(\sum_{\forall k} y_k \frac{\omega_k}{\min_k \omega_k} \right) - \right.$$

$$\left. \frac{1}{\psi} \cdot \left(\sum_{\forall k,m,w} a2c^k_{m,w} \cdot \frac{D^k_m}{D_w} + \sum_{\forall k,m,w} a2c^k_{m,w} \cdot \frac{M^k_m}{M_w} + \sum_{\forall k,i,j} l2e^k_{i,j} \cdot \frac{T^k_i}{T_j} \right) \right)$$

with ψ the cost normalization factor

$$\psi := \sum_{\forall k,m,w} \frac{D^k_m}{D_w} + \sum_{\forall k,m,w} \frac{M^k_m}{M_w} + \sum_{\forall k,i,j} \frac{T^k_i}{T_j}$$

The cost and revenue objective function maximizes an abstract representation of the revenue minus costs of the set of embedded NSIs. The revenue of the embedded NSIs is modeled as the sum of NSI weights as already used in the latency-aware objective function in Definition 17. The costs of network resource usage are represented by the the overall percentage usage of the resources. Consequently, if a resource is scarce, its percentage of utilization is higher than for an abundant resource. As a result, scarce resources are more expensive than abundant ones.
In this thesis, the cloud resources computation power D and memory M as well as the throughput T on the links are considered. The throughput costs for using substrate paths are a special case, since they have to be accounted per physical link included in the path. The percentage load of a virtual link on a substrate edge is added up on a per physical communication path basis.
When transforming the throughput term of the cost and revenue objective function using Definition 15, we get its l2p notation:

$$\sum_{\forall k,i,j} l2e^k_{i,j} \cdot \frac{T^k_i}{T_j}$$

$$\Leftrightarrow$$

$$\sum_{\forall k,i,j} \left(\sum_{\forall r} (l2p^k_{i,r} \cdot p2e_{r,j}) \right) \cdot \frac{T^k_i}{T_j}$$

Note that, free-self-links are omitted, since they do not cause costs by definition.

The resource costs are compared only relative to each other, since absolute values are often unavailable. The costs are normalized to 1 in order to not exceed the revenue of embedding any of the NSIs in the cost and revenue objective function.
For simplicity, it is assumed that embedding an NSI always provides a positive contribution margin. That means, at least the variable costs are compensated by the revenue of the NSI. As a consequence, embedding an additional NSI-R provides a positive financial gain for the MSP. As long as the NSI provisioning is priced reasonably, this is a realistic assumption because of the high share of fixed costs for the mobile network infrastructure and service provisioning.

4.3.4.4 Selecting the Objective Function

Selecting a suitable objective function is a crucial decision for the NSE. It does not only highly impact the selection of NSI-Rs that are accepted, but also their QoS and other parameters, for instance, the link latency, the resource costs and the profit of the MSPs and InPs. Beyond that, it impacts the runtime of solving the NSE problem.
Despotovic et al. show in [31] that the complexity of the objective function has a high impact on the runtime efficiency of the VNE problem. Choosing an appropriate objective function for a specific embedding problem is always a tradeoff between the accuracy of the implementation of the relevant objectives and the complexity of the objective function. Often, customized objective functions addressing the respective needs of the NSE problem at hand are needed. This might involve addressing additional goals beyond latency as well as cost and revenue optimization. Such objectives might regard, for instance, energy consumption or robustness of the resource provisioning. For instance, minimizing the energy consumption (this naturally induces energy cost reduction) is achieved by sharing physical resource among several virtual elements, for instance, several NSI applications might be deployed on the same cloud server instead of being distributed on several clouds. Then, unused network elements can be shut down in order to save electricity.
Moreover, a comprehensive approach for a robust, nearly optimal NSE is presented in Chapter 5.

4.3.5 Network Slice Embedding Models

Embedding an NSI into the mobile network infrastructure means to map every application node a^k_m on at least one cloud node c_w such that each virtual link l^k_i can be mapped on a suitable path P_r in the substrate and all resource requirements and network capabilities are fulfilled.

This section introduces three variants of the NSE Model, starting with the most basic one. The basic model is then extended to support path-splitting and MAI.

4.3.5.1 Basic Model

The basic NSE model is the simplest version of the nearly optimal NSE with ILP presented in this thesis.

Formalization The basic NSE model is an ILP formalization of the NSE embedding problem. It can be solved nearly optimally with an out-of-the-box solver. It considers end-to-end communication latency as well as link reliability and the most important link and node resources introduced in Section 4.3.2.
In addition, it allows so-called many-to-one mappings, i.e., several applications of the same or different network slices can be deployed on the same substrate node, if there are enough resources. The same applies to virtual link deployment. Several NSI communication links, regardless of their affiliation to the same or different NSIs can be deployed on the same substrate communication path.

The basic NSE model uses the simple objective function as defined in Definition 16. Nevertheless, a different objective function, especially one of the objective functions described in Section 4.3.4 can be used without the need to modify the remaining problem definition.
The objective function, in this case the simple objective function, is optimized under the following constraints, which describe the basic NSE problem as a standardized, mathematical ILP.
Four main classes of constraints can be distinguished:

- Capacity constraints
- Capability constraints
- Map-once constraints
- Graph constraints

Capacity constraints apply to node resources on the one hand and to link resources on the other hand. They ensure that the virtual elements of the NSIs can be mapped only on substrate elements which provide a sufficient amount of resources. The capacity constraints are defined similarly to the formalization proposed in [31].
For an arbitrary link resource R, we define the following set of constraints:

Definition 19 (General Link Resource Constraints)
The link resource constraints in the basic NSE model are defined as

$$\sum_{\forall k,i} l2e^k_{i,j} \cdot R^k_i \leq R_j,\ \forall j$$

The total required resources on all virtual links embedded on a substrate edge e_j must not exceed the provided resources of e_j.

With Definition 15, this translates to the following *l2p* mapping constraints.

$$\sum_{\forall k,i} \left(\sum_{\forall r} l2p^k_{i,r} \cdot p2e_{rj} \right) \cdot R^k_i \leq R_j, \, \forall j$$

Note that, free-self-links are not considered in the link resource constraints, since they do not consume any resources.
The link resource considered in this thesis is throughput only. Uplink and downlink resources are not distinguished. Beyond that, it is assumed that the provided as well as required throughput resources are constant for the basic NSE model.
Accordingly, the throughput resource constraints in the basic NSE model can be formalized as the throughput constraints:

Definition 20 (Throughput Constraints)
The throughput constraints in the basic NSE model are defined as

$$\sum_{\forall k,i} l2e^k_{i,j} \cdot T^k_i \leq T_j, \, \forall j$$

The capacity constraints state that the total throughput capacity of each link in the substrate network must not be exceeded by the sum of throughput requirements of all virtual links of the different NSIs mapped to it. This results in one constraint per physical link.
Transforming the above *l2e* based throughput constraints into *l2p* based constraints, using Definition 15, leads to the following set of constraints.

$$\sum_{\forall k,i} \left(\sum_{\forall r} l2p^k_{i,r} \cdot p2e_{r,j} \right) \cdot T^k_i \leq T_j, \, \forall j$$

Other possible link resources could be added to the model. In addition, uplink and downlink throughput could be distinguished.

Besides the link resources, node resources are taken into consideration. For an arbitrary node resource O we define the following set of constraints:

Definition 21 (General Node Resource Constraints)
The node resource constraints in the basic NSE model are defined as

$$\sum_{\forall k,m} a2c^k_{m,w} \cdot O^k_m \leq O_w, \, \forall w$$

For every NSI and every network slice link mapped on a cloud node c_w the required resources of the type O are summed up. The constraints ensure that the total amount of required resources must be smaller or equal than the provided resources on a per cloud node basis.
In this work, we consider computation power D and memory M resources. This results in the following sets of constraints.

Definition 22 (CPU Constraints)
The CPU constraints in the basic NSE model are defined as

$$\sum_{\forall k,m} a2c^k_{m,w} \cdot D^k_m \leq D_w, \, \forall w$$

The available computation power on the physical cloud nodes must not be exceeded by resource requirements of the assigned virtual application nodes.

Definition 23 (Memory Constraints)
The memory constraints in the basic NSE model are defined as

$$\sum_{\forall k,m} a2c^k_{m,w} \cdot M^k_m \leq M_w, \, \forall w$$

The same applies to the memory resources of the cloud and application nodes.

Beyond the capacity constraints, the NSE is subject to capability constraints regarding, for instance, the end-to-end latency as well as the node and link reliability.

We define a latency constraint for every embedded network slice link:

Definition 24 (Latency Constraints)
The end-to-end latency constraints in the basic NSE model are defined as

$$l2p^k_{i,r} \cdot L_{P_r} \leq L^k_i, \, \forall k, i, r$$

The allowed latency of each virtual link in any of the NSIs must be greater or equal to the sum of the actual latencies of all physical edges contained in the path mapped to this virtual link.
Broken down to $l2e$ mappings, using Definition 15, the latency constraints can be denoted as:

$$\sum_{\forall j} l2e^k_{i,j} \cdot p2e_{r,j} \cdot L_j \leq L^k_i, \, \forall k, i, r$$

In this basic NSE model, latency is assumed to be constant. If the communication links are not overloaded it is feasible to define an upper bound for the actual latency on a physical link.

The capability constraints include the so-called quality constraints for the links and nodes. The binary and qualitative link quality constraints are defined as follows.

Definition 25 (Network Link Quality Constraints)
The link quality constraints in the basic NSE model are defined as

$$l2e^k_{i,j} \cdot Q^k_i \leq Q_j, \quad \forall k, i, j$$

If Q is binary, then the constraints are called binary link quality constraints.
If Q has an infinite value range, then the constraints are called quantitative link quality constraints.

The constraints verify if a specific required capability Q is available on the physical link the virtual link has been mapped on. There is one constraint for each $l2e$ variable, that means, for each combination of a virtual and a physical link across all NSIs. This can be applied to binary parameters as well as parameter with continuous values.
An example for a binary constraint is restricting a virtual communication link in the RAN to a specified RAN technology, like LTE only. In this case, Q_i^k is set to 1 if and only if the associated i-th link of the k-th network slice l_i^k requires LTE. Otherwise Q_i^k is set to 0. Similarly, Q_j is set to 1 if and only if the associated j-th edge of the substrate network is an LTE connection. Otherwise Q_j is set to 0. The constraints ensure that l_i^k can only be mapped on an edge e_j if Q_i^k is smaller or equal Q_j. That means, if LTE is required, it has to be provided on the allocated link. If LTE is not required, it does not have to be provided by the used link.
Quantitative constraints use infinite value ranges for Q, for instance, the reliability of a communication link represented by a percentage. Q can take values from the interval $[0, 1]$. The quantitative link quality constraint checks, whether or not the required reliability is provided by the physical communication link a network slice link is mapped on.

From an $l2p$ mapping perspective, the general link quality constraints can be formalized as follows.

$$l2p_{i,r}^k \cdot Q_i^k \leq Q_{P_r}, \quad \forall k, i, r$$

I.e., the minimum quality aggregated with respect to the physical path P_r must fulfill the requirement of the virtual link l_i^k mapped on it.

For the basic NSE model, only the link reliability constraints are included. They are modeled as follows.

Definition 26 (Link Reliability Constraints)
The link reliability constraints in the basic NSE model are defined as

$$l2e_{i,j}^k \cdot A_i^k \leq A_j, \quad \forall k, i, j$$

For the required reliability of the virtual links, the actual reliability of all physical links e_j used in a path mapped to a virtual link l_i^k must fulfill the requirements of the link. However, the reliability restrictions for link mappings do only consider the reliability of the physical edges on the path, but not of the cloud server nodes visited on that path.
The link reliability constraints in Definition 26 transferred into the $l2p$ mapping notation are similar to the general link quality constraints and look as follows.

$$l2p_{i,r}^k \cdot A_i^k \leq A_{P_r}, \quad \forall k, i, r$$

Binary and quantitative node quality constraints are similarly defined for the vertices or nodes in the NSE problem.

Definition 27 (Network Node Quality Constraints)
The node quality constraints in the basic NSE model are defined as

$$a2c^k_{m,w} \cdot W^k_m \leq W_w, \quad \forall k, m, w$$

If W is binary, then the constraints are called binary node quality constraints. If W has a continuous value range, then the constraints are called quantitative node quality constraints.

The node mapping must comply with the required quality attributes. For instance, virtual applications can only be mapped on physical could servers which comply with their minimum reliability.

Definition 28 (Network Node Reliability Constraints)
The node reliability constraints in the basic NSE model are defined as

$$a2c^k_{m,w} \cdot B^k_m \leq B_w, \quad \forall k, m, w$$

There is one network node reliability constraint for each application to cloud node mapping.

Furthermore, the basic NSE model specifies the following so-called map nodes once constraints. They are responsible for ensuring that every virtual application node is deployed exactly once in the substrate.

Definition 29 (Map Nodes Once Constraints)
The map nodes once constraints in the basic NSE model are defined as

$$\sum_{\forall w} a2c^k_{m,w} = y_k, \, \forall k, m$$

The map nodes once constraints are similarly defined as in the ILP formulation of the VNetMapper in [31]. There is one constraint for each application node in each network slice. If the k-th NSI-R is accepted, then $y_k = 1$ and all application nodes a^k_m of the k-th NSI-R must be embedded exactly once in the substrate network. The left-hand side of the map-once constraints sums up the values of all $a2c^k_{m,w}$ variables for all cloud nodes c_w in the substrate, for a fixed k and a fixed m. That means, for a specific application node a^k_m. Thus, this sum reflects the number of embeddings of the m-th application of the k-th network slice in the substrate. It must be equal to 1 if the NSI-R is accepted and equal to 0 if the NSI-R is rejected.

The NSE model requires, that each virtual link is mapped exactly once.

Definition 30 (Map Links Once Constraints)
The map links once constraints in the basic NSE model are defined as

$$\sum_{\forall r} l2p^k_{i,r} = y_k, \, \forall k, i$$

If the k-th NSI-R is accepted, that means, if $y_k = 1$, then every virtual link l_i^k of this NSI must be mapped on exactly one physical communication path P_r. Otherwise, if the k-th NSI-R is not embedded, then none of its associated virtual links is mapped on a physical communication path in the substrate.

In addition, two types of graph constraints regarding the link mapping are needed in the basic NSE model. They ensure, that the virtual links are mapped on physical paths that connect the two cloud nodes, its adjacent applications are mapped on, or the connected UE, respectively. This is achieved by the link mapping constraints in the Definitions 31 and 32.

Definition 31 (Map Adjacent Links Constraints)
The map adjacent links constraints in the basic NSE model are defined as

$$\sum_{\forall P_r \in \mathcal{P}_{c_v c_w}} l2p_{i,r}^k \geq a2c_{m,w}^k \ ,\forall k,i,w,m \ \textit{if} \ l_i^k = \{a_m^k, a_b^k\}$$

If an NSI application a_m^k is mapped on a cloud node c_w in the substrate network, then every link connected to a_m^k in the NSI specification must be mapped on at least one path P_r with $P_r \in \mathcal{P}_{c_v c_w}$ with $\mathcal{P}_{c_v c_w} = \mathcal{P}_{c_w c_v}$, that means, a path which has c_w as one of its end-nodes.
The set of equations defined in Definition 31 also excludes any link to path mappings that are not matching the node mapping, as long as two applications are connected. The same is achieved for the UE group to application links with the equations in definition Definition 32.

Definition 32 (Map UE Links Constraints)
The map UE links constraints in the basic NSE model are defined as

$$\sum_{\forall P_r \in \mathcal{P}_{u_v c_w}} l2p_{i,r}^k = y_k \ ,\forall k,i \ \textit{if} \ l_i^k = \{u_v, a_b^k\}$$

If the k-th NSI-R is accepted for deployment in the physical network infrastructure, i.e., if $y_k = 1$, then for every link l_i^k with an adjacent UE node u_v exactly one link to path mapping $l2p_{i,r}^k$ on a path with u_v as one of its end-nodes must exist. Otherwise, if the k-th NSI-R is not mapped, no such mapping must exist.

In addition, the links connecting the UEs and the applications, must be mapped corresponding to the application node mapping in the substrate network.
This is achieved by the following set of constraints:

Definition 33 (UE to Application Links Graph Constraints)
The UE to application links graph constraints in the basic NSE model are defined as

$$a2c_{m,w}^k \geq l2p_{i,r}^k, \ \forall k,i,r \ \textit{with} \ l_i^k = \{u_v, a_m^k\} \ \textit{and} \ P_r \in \mathcal{P}_{c_v,c_w}$$

If $l_i^k = \{u_v, a_m^k\}$ and l_i^k is mapped on $P_r \in \mathcal{P}_{c_v,c_w}$ with $\mathcal{P}_{c_v,c_w} = \mathcal{P}_{c_w,c_v}$ with the adjacent nodes u_v and c_w, then a_m^k must be mapped on c_w. Nevertheless, it is possible that a_m^k is mapped on c_w, i.e., $a2c_{m,w}^k = 1$ but l_i^k has not been assigned to the path $P_r \in \mathcal{P}_{c_v,c_w}$.

Further restrictions are required to ensure that only feasible combinations of node and link mappings are accepted. Incoming and outgoing links must be suitable. This is achieved with the map adjacent nodes constraints in Definition 34.

Definition 34 (Map Adjacent Nodes Constraints)
The map adjacent nodes constraints in the basic NSE model are defined as

$$a2c_{m,v}^k + a2c_{m,w}^k \geq l2p_{i,r}^k,\ \forall k, i, r \textit{ with } l_i^k = \{a_m^k, a_q^k\} \textit{ and } P_r \in \mathcal{P}_{c_v,c_w}$$

and

$$a2c_{q,v}^k + a2c_{q,w}^k \geq l2p_{i,r}^k,\ \forall k, i, r \textit{ with } l_i^k = \{a_m^k, a_q^k\} \textit{ and } P_r \in \mathcal{P}_{c_v,c_w}$$

if both ends of the path are cloud nodes, or

$$a2c_{q,w}^k \geq l2p_{i,r}^k,\ \forall k, i, r \textit{ with } l_i^k = \{u_v, a_q^k\} \textit{ and } P_r \in \mathcal{P}_{u_v,c_w}$$

if one end of the path is a UE group node, or

$$a2c_{m,v}^k \geq l2p_{i,r}^k,\ \forall k, i, r \textit{ with } l_i^k = \{a_m^k, a_q^k\} \textit{ and } P_r = \mathcal{P}_{c_v}^{free}$$

and

$$a2c_{q,v}^k \geq l2p_{i,r}^k,\ \forall k, i, r \textit{ with } l_i^k = \{a_m^k, a_q^k\} \textit{ and } P_r = \mathcal{P}_{c_v}^{free}$$

if P_r is a free-self-path.

The set of inequations in Definition 34 concern the link mapping, if $l_i^k = \{a_m^k, a_q^k\}$ and l_i^k is mapped on $P_r \in \mathcal{P}_{c_v,c_w}$ with $\mathcal{P}_{c_v,c_w} = \mathcal{P}_{c_w,c_v}$ and with the cloud nodes c_v and c_w as its end-nodes, then a_m^k must be mapped on c_w or c_v and a_q^k must be mapped on c_v or c_w or vice versa.
Note that, the inequations in Definition 34 do not exclude that both ends of the virtual link a_m^k and a_q^k are mapped on the same physical end-node of P_r. That means, both could be mapped on c_w or both on c_v. However, this is prevented by the inequations in Definition 31.
If $l_i^k = \{u_v, a_q^k\} = \{a_q^k, u_v\}$ is mapped on a path $P_r \in \mathcal{P}_{u_v,c_w}$ with $\mathcal{P}_{u_v,c_w} = \mathcal{P}_{c_w,u_v}$, i.e., on a path with a UE group node as one of its end-nodes, then a_q^k must be mapped on c_w.
Another special case is, when a virtual link $l_i^k = \{a_m^k, a_q^k\}$ is mapped on a free-self-path $P_r = P_{c_v}^{free}$. Then a_m^k as well as a_q^k have to be mapped on c_v.

This model allows to share the same cloud node among chained applications. This is achieved by allowing paths to start in an arbitrary cloud server node (not only in a UE node) and by adding paths that only consist of one free-self link, as defined in Definition 6), i.e., for every cloud node $c_w \in \mathcal{C}$ of the substrate network a path is defined which only consists of the free-self-link of c_w. Similarly to the underlying free-self-links, these free-self-paths have zero latency, infinite throughput resources and a reliability of 1.

Summary of the Basic NSE Model When the components introduced above are combined, the basic NSE model can be summarized as follows.

Model 1 (Basic NSE Model)

$$\max_{y_k} \left(\sum_{\forall k} y_k \frac{\omega_k}{\sum_{\forall k} \omega_k} \right)$$

under

1. *Throughput Constraints:*
$$\sum_{\forall k,i} l2e^k_{i,j} \cdot T^k_i \leq T_j,\ \forall j$$
2. *CPU Constraints:*
$$\sum_{\forall k,m} a2c^k_{m,w} \cdot D^k_m \leq D_w,\ \forall w$$
3. *Memory Constraints:*
$$\sum_{\forall k,m} a2c^k_{m,w} \cdot M^k_m \leq M_w,\ \forall w$$
4. *Latency Constraints:*
$$l2p^k_{i,r} \cdot L_{P_r} \leq L^k_i,\ \forall k, i, r$$
5. *Link Reliability Constraints:*
$$l2e^k_{i,j} \cdot A^k_i \leq A_j, \quad \forall k, i, j$$
6. *Network Node Reliability Constraints:*
$$a2c^k_{m,w} \cdot B^k_m \leq B_w, \quad \forall k, m, w$$
7. *Map Nodes Once Constraints:*
$$\sum_{\forall w} a2c^k_{m,w} = y_k,\ \forall k, m$$
8. *Map Links Once Constraints:*
$$\sum_{\forall r} l2p^k_{i,r} = y_k,\ \forall k, i$$
9. *Map Adjacent Links Constraints:*
$$\sum_{\forall P_r \in \mathcal{P}_{c_v,c_w}} l2p^k_{i,r} \geq a2c^k_{m,w}\ ,\forall k, i, w, m\ \textit{if}\ l^k_i = \{a^k_m, a^k_b\}$$

10. Map UE Links Constraints:

$$\sum_{\forall P_r \in \mathcal{P}_{u_v,c_w}} l2p^k_{i,r} = y_k \ ,\forall k, i, \ \textit{if} \ l^k_i = \{u_v, a^k_b\}$$

11. UE to Application Links Graph Constraints:

$$a2c^k_{m,w} \geq l2p^k_{i,r}, \ \forall k, i, r \ \textit{with} \ l^k_i = \{u_v, a^k_m\} \ \textit{and} \ P_r \in \mathcal{P}_{c_v,c_w}$$

12. Map Adjacent Nodes Constraints:

a) for Cloud Node to Cloud Node Paths:

$$a2c^k_{m,v} + a2c^k_{m,w} \geq l2p^k_{i,r}, \ \forall k, i, r \ \textit{with} \ l^k_i = \{a^k_m, a^k_q\} \ \textit{and} \ P_r \in \mathcal{P}_{c_v,c_w} \ \textit{and}$$

$$a2c^k_{q,v} + a2c^k_{q,w} \geq l2p^k_{i,r}, \ \forall k, i, r \ \textit{with} \ l^k_i = \{a^k_m, a^k_q\} \ \textit{and} \ P_r \in \mathcal{P}_{c_v,c_w}$$

b) for UE Group Node to Cloud Node Paths:

$$a2c^k_{q,w} \geq l2p^k_{i,r}, \ \forall k, i, r \ \textit{with} \ l^k_i = \{u_v, a^k_q\} \ \textit{and} \ P_r \in \mathcal{P}_{u_v,c_w}$$

c) for Free-Self-Paths:

$$a2c^k_{m,v} \geq l2p^k_{i,r}, \ \forall k, i, r \ \textit{with} \ l^k_i = \{a^k_m, a^k_q\} \ \textit{and} \ P_r = \mathcal{P}^{free}_{c_v} \ \textit{and}$$

$$a2c^k_{q,v} \geq l2p^k_{i,r}, \ \forall k, i, r \ \textit{with} \ l^k_i = \{a^k_m, a^k_q\} \ \textit{and} \ P_r = \mathcal{P}^{free}_{c_v}$$

The simple objective function used in the description of the basic NSE model can be replaced by any of the objective function described in Section 4.3.4 without modification of the constraints.

Algorithmic Details The basic NSE model requires to determine all cycle-free physical communication paths as defined in Definition 5. These paths are detected by running a recursive search.
The physical communication path finder (see Algorithm 1) creates a list of physical cycle-free communication paths. For every UE group and cloud node in the substrate network, the according free-self path is added to the list of physical cycle-free paths. Then the recursive depth first search method in Algorithm 2 is executed on every node.
Algorithm 2 takes the current node (in the beginning this is the initial UE group node), the list of already created (finished) paths $\mathcal{P}_f$ (which is empty in the beginning) as well as the unfinished path P, that is currently created (which has no elements in the first execution of the method). Then a depth-first search starting from the current node is executed. Every outgoing edge is analyzed and according paths are added to the list of finished paths, if these paths are not in the list (also not in reverse order) and the added link does not close a cycle. Furthermore, free-self-links are excluded (see line 3). In addition to that, the algorithm does not add paths which include more than one UE group node. UE group nodes are not used as relays in paths. When a new cloud node has been reached in the graph, the depth

Algorithm 1 Physical Communication Path Finder

```
1: create an empty list of finished paths $\mathcal{P}_f$
2: for each node $d \in \mathcal{U} \cup \mathcal{C}$ do
3:    add free-self path $P_d^{free}$ to $\mathcal{P}_f$
4:    create empty unfinished path $P$
5:    Depth First Search($d$, $\mathcal{P}_f$, $P$)
6: end for
```

Algorithm 2 Depth First Search

```
1: Input: current node $d_0$, list of finished paths $\mathcal{P}_f$, current unfinished path $P$
2: for each edge $e = d_0d_1$ adjacent to $d_0$ do
3:    if $d_0 \neq d_1$ then
4:       append $e$ to $P$
5:       if $P$ is not a cycle then
6:          if not both end-nodes of $P$ are UE nodes then
7:             add $P$ to $\mathcal{P}_f$
8:             if $d_1$ is a cloud node then
9:                Depth First Search($d_1$, $\mathcal{P}_f$, $P$)
10:            end if
11:         end if
12:      end if
13:   end if
14:   if $e$ has been appended to $P$ then
15:      remove $e$ from $P$
16:   end if
17: end for
```

first search algorithm is recursively executed with this new node d_1, the current list of finished paths and the current unfinished Path P. A further recursive search is prevented by the if-statement in line 8, once a UE group node is reached. Lines 14 and 15 reset the currently created physical communication path P for the next iteration of the for-loop, starting in line 2.
The structure of the substrate network as well as the NSIs brings about, that not all $l2p$ mappings are feasible. If the virtual link l_i^k connects a UE group with an application node, then this link can only be mapped on physical paths which have this UE group as one of its end-nodes. In reverse, if a physical path has a UE group as one of its end-nodes, this UE group must be adjacent to the associated virtual link. This implies, that an $l2p$ mapping is infeasible, if the virtual link is adjacent to a UE group which is not an end-node of the physical link and vice versa. These $l2p$ mapping variables cannot take a value different to zero. Therefore, these variables are omitted in the practical implementation. This reduces the set of variables of the NSE Model to the feasible set of variables, which can improve the runtime efficiency of the solver.

Example - Basic Model A simple example is used to demonstrate the NSE Models. The substrate network (see Figure 4.4) comprises two UE groups connected to an edge cloud c_0 and a central cloud node c_1.

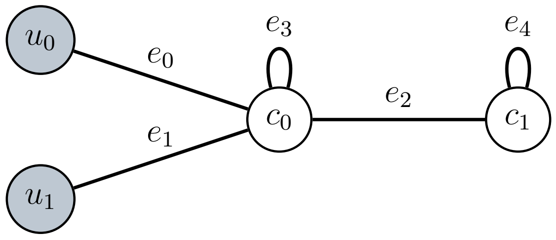


Figure 4.4: Model Example - Substrate Network

In addition, two NSI-Rs, depicted in Figures 4.5 and 4.6, are given.

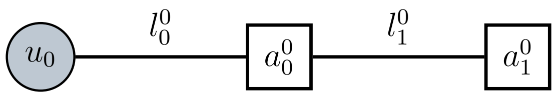


Figure 4.5: Model Example - Network Slice 0

The resources and capabilities provided by the substrate elements and required by the NSI-Rs elements are summarized in the Tables 4.4 and 4.5.
This graphically represented NSE problem is now formalized as an ILP using the basic NSE model.
First of all, the variables are defined.
One binary embedding variable for each NSI-R is defined, i.e., y_0 and y_1.
In addition, for each NSI the binary application to cloud node mapping variables are defined (all possible combinations), see Table 4.6.

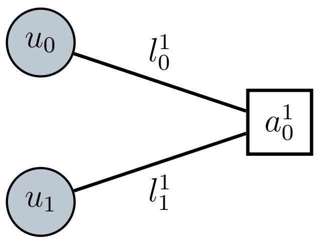


Figure 4.6: Model Example - Network Slice 1

Table 4.4: Model Example - Network Node Parameters

Node	**CPU**	**Memory**	**Reliability**
c_0	12	10	0.95
c_1	80	100	0.99
a_0^0	5	7	0.92
a_1^0	7	2	0.93
a_0^1	4	2	0.94

Table 4.5: Model Example - Network Link Parameters

Link	**Throughput**	**Latency**	**Reliability**
e_0	5	1	0.95
e_1	5	2	0.95
e_2	10	1	0.95
e_3	∞	0	1.00
e_4	∞	0	1.00
l_0^0	2	1	0.90
l_1^0	2	1	0.90
l_0^1	1	3	0.90
l_1^1	1	3	0.90

Table 4.6: Model Example - Application to Cloud Node Mapping Variables

	a_0^0	a_1^0	a_0^1
c_0	$a2c_{0,0}^0$	$a2c_{1,0}^0$	$a2c_{0,0}^1$
c_1	$a2c_{0,1}^0$	$a2c_{1,1}^0$	$a2c_{0,1}^1$

For example, the value of $a2c^0_{0,0}$ determines whether the application a^0_0 is mapped on the cloud node c_0 or not.
In the substrate network, the follow physical communication paths are defined:

$$
\begin{aligned}
&P_0 := \{e_0\}\\
&P_1 := \{e_0, e_2\}\\
&P_2 := \{e_1\}\\
&P_3 := \{e_1, e_2\}\\
&P_4 := P^{free}_{c_0} = \{e_3\}\\
&P_5 := \{e_2\}\\
&P_6 := P^{free}_{c_1} = \{e_4\}
\end{aligned}
$$

These paths are determined by running a recursive search starting with the list of all substrate nodes (UE group nodes and cloud nodes), see Algorithm 1. For every cloud node, the corresponding free-self-path is created. From UE u_0, the vertex e_0 is detected, forming the first path P_0. c_0 is added to the node list and $P_4 = P^{free}_{c_0}$ is created. The Depth-First-Search also detects e_2 and creates the path P_1. Then c_1 is added to the node list and $P_6 = P^{free}_{c_1}$ is created. Starting from u_1, the paths P_2 and P_3 are detected. From c_0 there is only one undetected path, consisting of the link e_2, left. It is named P_5. At c_1 no new paths are found.

Furthermore, the link to path mapping variables shown in Table 4.7 are defined.

Table 4.7: Model Example - Link to Path Mapping Variables

	l^0_0	l^0_1	l^1_0	l^1_1
P_0	$l2p^0_{0,0}$	-	$l2p^1_{0,0}$	-
P_1	$l2p^0_{0,1}$	-	$l2p^1_{0,1}$	-
P_2	-	-	-	$l2p^1_{1,2}$
P_3	-	-	-	$l2p^1_{1,3}$
P_4	-	$l2p^0_{1,4}$	-	-
P_5	-	$l2p^0_{1,5}$	-	-
P_6	-	$l2p^0_{1,6}$	-	-

Without loss of generality, only feasible $l2p$ mappings are used in the actual implementation of the NSE algorithms. I.e., only the $l2p$ variables with suitable end-nodes are used. For instance, the path P_1 consists of the edges e_0, e_1 and the end-nodes u_0 and c_1. Therefore, only the links l^0_0 and l^1_0 can be feasibly mapped on P_0, since they are the only virtual communication links adjacent to u_0 in the two NSIs. Another interesting example is the path P_5 which consists of e_2 with the end-nodes c_0 and c_1. Consequently, only virtual links connecting two application nodes (no UE group nodes) can be mapped on P_5. In this example, only l^0_1 is such a feasible candidate.

Secondly, the simple objective-function for the basic NSE model is created. Note that, in this practical implementation only the variables that have a feasible non-zero value are considered in the objective function and the constraints. This improves the runtime-efficiency of the algorithm, without loss of generality.
Since the NSI weights are defined equally as $\omega_0 = \omega_1 = 1$, the simple objective function in this example reduces to:

$$\max_{y_k} \frac{1 \cdot y_0 + 1 \cdot y_1}{1+1}$$

$$\Longleftrightarrow$$

$$\max_{y_k}(0.5 \cdot y_0 + 0.5 \cdot y_1)$$

Third, the constraints for this simple example are listed below. Only the feasible embeddings are considered in this implementation.

1. Throughput Constraints:
One throughput capacity constraint for each edge in the substrate network is defined.

- Throughput capacity constraint for e_0:

  $2 \cdot l2p^0_{0,0} + 2 \cdot l2p^0_{0,1} + 1 \cdot l2p^1_{0,0} + 1 \cdot l2p^1_{0,1} + \leq 5$

- Throughput capacity constraint for e_1:

  $1 \cdot l2p^1_{1,2} + 1 \cdot l2p^1_{1,3} \leq 5$

- Throughput capacity constraint for e_2:

  $2 \cdot l2p^0_{0,1} + 1 \cdot l2p^1_{0,1} + 1 \cdot l2p^1_{1,3} + 2 \cdot l2p^0_{1,5} \leq 10$

The node resource constraints are defined for the CPU as well as the memory capacity of each substrate node.

2. CPU Constraints:

- CPU capacity constraint for cloud node c_0:

  $5 \cdot a2c^0_{0,0} + 7 \cdot a2c^0_{1,0} + 4 \cdot a2c^1_{0,0} \leq 12$

- CPU capacity constraint for cloud node c_1:

  $5 \cdot a2c^0_{0,1} + 7 \cdot a2c^0_{1,1} + 4 \cdot a2c^1_{0,1} \leq 80$

3. Memory Constraints:

- Memory capacity constraint for cloud node c_0:

  $7 \cdot a2c^0_{0,0} + 2 \cdot a2c^0_{1,0} + 2 \cdot a2c^1_{0,0} \leq 10$

- Memory capacity constraint for cloud node c_1:

  $7 \cdot a2c^0_{0,1} + 2 \cdot a2c^0_{1,1} + 2 \cdot a2c^1_{0,1} \leq 100$

4. Latency Constraints:
The communication latency of every path (sum of latencies of the elements of the physical path) must not exceed the allowed latency of the virtual link. That means, the sum of the latencies of the used edges in the path must be less or equal to the virtual link latency.

- Link latency constraint for virtual link l^0_0 and:

  P_0: $1 \cdot l2p^0_{0,0} \leq 1 \Leftrightarrow 1 \cdot l2e^0_{0,0} \leq 1$

  P_1: $2 \cdot l2p^0_{0,1} \leq 1 \Leftrightarrow 1 \cdot l2e^0_{0,0} + 1 \cdot l2e^0_{0,2} \leq 1$

- Link latency constraint for virtual link l^0_1 and:

  P_4: $0 \cdot l2p^0_{1,4} \leq 1 \Leftrightarrow 0 \cdot l2e^0_{1,3} \leq 1$

  P_5: $1 \cdot l2p^0_{1,5} \leq 1 \Leftrightarrow 1 \cdot l2e^0_{1,2} \leq 1$

  P_6: $0 \cdot l2p^0_{1,6} \leq 1 \Leftrightarrow 0 \cdot l2e^0_{1,4} \leq 1$

- Link latency constraint for virtual link l^1_0 and:

  P_0: $1 \cdot l2p^1_{0,0} \leq 3 \Leftrightarrow 1 \cdot l2e^1_{0,0} \leq 3$

  P_1: $2 \cdot l2p^1_{0,1} \leq 3 \Leftrightarrow 1 \cdot l2e^1_{0,0} + 1 \cdot l2e^1_{0,2} \leq 3$

- Link latency constraint for virtual link l^1_1 and:

  P_2: $2 \cdot l2p^1_{1,2} \leq 3 \Leftrightarrow 2 \cdot l2e^1_{1,1} \leq 3$

  P_3: $3 \cdot l2p^1_{1,3} \leq 3 \Leftrightarrow 2 \cdot l2e^1_{1,1} + 1 \cdot l2e^1_{1,2} \leq 3$

5. Link Reliability Constraints:
A virtual link can only be mapped on a suitable physical path, if the path fulfills its reliability requirements.

- Link reliability constraint for virtual link l^0_0 and:

  P_0: $0.9 \cdot l2p^0_{0,0} \leq 0.95 \Leftrightarrow 0.9 \cdot l2e^0_{0,0} \leq 0.95$

  P_1: $0.9 \cdot l2p^0_{0,1} \leq 0.95 \Leftrightarrow (0.9 \cdot l2e^0_{0,0} \leq 0.95) \wedge (0.9 \cdot l2e^0_{0,2} \leq 0.95)$

- Link reliability constraint for virtual link l_1^0 and:

  P_4: $0.9 \cdot l2p_{1,4}^0 \leq 0.95 \Leftrightarrow 0.9 \cdot l2e_{1,3}^0 \leq 0.95$

  P_5: $0.9 \cdot l2p_{1,5}^0 \leq 0.95 \Leftrightarrow 0.9 \cdot l2e_{1,2}^0 \leq 0.95$

  P_6: $0.9 \cdot l2p_{1,6}^0 \leq 0.95 \Leftrightarrow 0.9 \cdot l2e_{1,4}^0 \leq 0.95$

- Link reliability constraint for virtual link l_0^1 and:

  P_0: $0.9 \cdot l2p_{0,0}^1 \leq 0.95 \Leftrightarrow 0.9 \cdot l2e_{0,0}^1 \leq 0.95$

  P_1: $0.9 \cdot l2p_{0,1}^1 \leq 0.95 \Leftrightarrow (0.9 \cdot l2e_{0,0}^1 \leq 0.95) \wedge (0.9 \cdot l2e_{0,2}^1 \leq 0.95)$

- Link reliability constraint for virtual link l_1^1 and:

  P_2: $0.9 \cdot l2p_{1,2}^1 \leq 0.95 \Leftrightarrow 0.9 \cdot l2e_{1,1}^1 \leq 0.95$

  P_3: $0.9 \cdot l2p_{1,3}^1 \leq 0.95 \Leftrightarrow (0.9 \cdot l2e_{1,1}^1 \leq 0.95) \wedge (0.9 \cdot l2e_{1,2}^1 \leq 0.95)$

6. Network Node Reliability Constraints:
An application can only be mapped on a cloud node, if the cloud node fulfills its reliability requirements.

$a2c_{0,0}^0 \cdot 0.92 \leq 0.95$

$a2c_{1,0}^0 \cdot 0.93 \leq 0.95$

$a2c_{0,0}^1 \cdot 0.94 \leq 0.95$

$a2c_{0,1}^0 \cdot 0.92 \leq 0.99$

$a2c_{1,1}^0 \cdot 0.93 \leq 0.99$

$a2c_{0,1}^1 \cdot 0.94 \leq 0.99$

7. Map Nodes Once Constraints:
Each application is embedded exactly once into the substrate network.

a_0^0: $a2c_{0,0}^0 + a2c_{0,1}^0 = y_0$

a_1^0: $a2c_{1,0}^0 + a2c_{1,1}^0 = y_0$

a_0^1: $a2c_{0,0}^1 + a2c_{0,1}^1 = y_1$

8. Map Links Once Constraints:
Every virtual link is embedded exactly once into the substrate network.

l_0^0: $l2p_{0,0}^0 + l2p_{0,1}^0 = y_0$

l_1^0: $l2p_{1,4}^0 + l2p_{1,5}^0 + l2p_{1,6}^0 = y_0$

l_0^1: $l2p_{0,0}^1 + l2p_{0,1}^1 = y_1$

l_1^1: $l2p_{1,3}^1 + l2p_{1,2}^1 = y_1$

9. Map Adjacent Links Constraints:
All virtual links connecting two applications must be mapped on a suitable path with regard to the application to cloud node mapping. In this simple example the only affected link is l_1^0.

$l2p_{1,4}^0 + l2p_{1,5}^0 \geq a2c_{0,0}^0$

$l2p_{1,6}^0 + l2p_{1,5}^0 \geq a2c_{0,1}^0$

$l2p_{1,4}^0 + l2p_{1,5}^0 \geq a2c_{1,0}^0$

$l2p_{1,6}^0 + l2p_{1,5}^0 \geq a2c_{1,1}^0$

10. Map UE Links Constraints:
If a network slice is embedded all outgoing links of the UE group must be mapped on a suitable physical path.

l_0^0: $l2p_{0,0}^0 + l2p_{0,1}^0 = y_0$

l_0^1: $l2p_{0,0}^1 + l2p_{0,1}^1 = y_1$

l_1^1: $l2p_{1,2}^1 + l2p_{1,3}^1 = y_1$

11. UE to Application Links Graph Constraints:
All virtual links must be mapped suitably.

l_0^0: $a2c_{0,0}^0 \geq l2p_{0,0}^0$ and $a2c_{0,1}^0 \geq l2p_{0,1}^0$

l_0^1: $a2c_{0,0}^1 \geq l2p_{0,0}^1$ and $a2c_{0,1}^1 \geq l2p_{0,1}^1$

l_1^1: $a2c_{0,0}^1 \geq l2p_{1,2}^1$ and $a2c_{0,1}^1 \geq l2p_{1,3}^1$

12. Map Adjacent Nodes Constraints:
The application node mapping must fit to the virtual link mapping.

For the start-node mapping:

Start-node of path P_4 if mapped on l_1^0: $a2c_{0,0}^0 \geq l2p_{1,4}^0$

Start-node of path P_5 if mapped on l_1^0: $a2c_{0,0}^0 + a2c_{0,1}^0 \geq l2p_{1,5}^0$

Start-node of path P_6 if mapped on l_1^0: $a2c_{0,1}^0 \geq l2p_{1,6}^0$

For the end-node mapping:

End of path P_0 if mapped on l_0^0: $a2c_{0,0}^0 \geq l2p_{0,0}^0$

End-node of path P_1 if mapped on l_0^0: $a2c_{0,1}^0 \geq l2p_{0,1}^0$

End-node of path P_4 if mapped on l_1^0: $a2c_{1,0}^0 \geq l2p_{1,4}^0$

End-node of path P_5 if mapped on l_1^0: $a2c_{1,0}^0 + a2c_{1,1}^0 \geq l2p_{15}^0$

End-node of path P_6 if mapped on l_1^0: $a2c_{1,1}^0 \geq l2p_{1,6}^0$

End-node of path P_0 if mapped on l_0^1: $a2c_{0,0}^1 \geq l2p_{0,0}^1$

End-node of path P_1 if mapped on l_0^1: $a2c_{0,1}^1 \geq l2p_{0,1}^1$

End-node of path P_2 if mapped on l_1^1: $a2c_{0,0}^1 \geq l2p_{1,2}^1$

End-node of path P_3 if mapped on l_1^1: $a2c_{0,1}^1 \geq l2p_{1,3}^1$

This ILP can be solved with an out-of-the-box solver using the model introduced in Section 2.3.5.
The result is the variable assignment which maximizes the objective function and respects all constraints. Note that the solvers do not always provide the optimal solutions, but only a nearly optimal solution, since they apply heuristics.

In the best solution of the simple NSE example, the following variables are set to the value 1. All other variables, not explicitly mentioned here take the value 0.

$y_0 = 1$	$a2c_{0,0}^0 = 1$	$l2p_{0,0}^0 = 1$
$y_1 = 1$	$a2c_{1,0}^0 = 1$	$l2p_{1,4}^0 = 1$
	$a2c_{0,1}^1 = 1$	$l2p_{0,1}^1 = 1$
		$l2p_{1,3}^1 = 1$

Finally, this variable assignment is translated back into an NSI embedding and resource allocation solution. Since $y_0 = y_1 = 1$ both NSI-Rs are accepted and embedded into the substrate network. The nodes and links are mapped as illustrated in Figure 4.7.

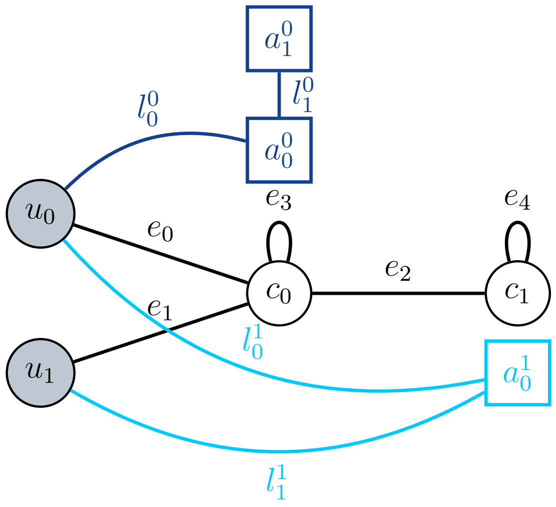


Figure 4.7: Basic Model Example - Simple Optimization

The physical nodes c_0 has a CPU capacity of 12 entities and a memory capacity of 10 entities, while its reliability is given with 95%. This allows to map an arbitrary combination of two of the three virtual application nodes from the two NSI-Rs. Beyond that, the virtual link l_0^0 has a maximum latency of 1 and therefore it can only be deployed on the path P_0. Consequently, a_0^0 must be mapped on c_0. The throughput and link reliability constraints of l_0^0 are fulfilled by P_0. The second virtual link of NSI 0 l_1^0 can either be allocated on the free-self-path P_4 or on P_5. Both paths provide the required resources and capabilities on the link and both cloud nodes (c_0 and c_1) can accommodate a_1^0. The basic NSE model with the simple objective function makes a random choice for the mapping of l_1^0 together with the associated allocation of a_1^0. Similarly, a_0^1 can be mapped on both cloud nodes. The required link resources and capabilities for the corresponding link mapping are available.

The objective function used for the basic NSE model can be easily exchanged. In the following the simple NSE example is solved with the latency-aware as well as the cost and revenue objective function.

The latency-aware objective function for the simple NSE example is determined as follows.

$$\max_{y_k, l2p_{ir}^k} \left(\frac{1 \cdot y_0 + 1 \cdot y_1}{\min\{1, 1\}} \right.$$

$$- \frac{l2p_{0,0}^0 \cdot L_{P_0} + l2p_{0,1}^0 \cdot L_{P_1} + l2p_{1,4}^0 \cdot L_{P_4} + l2p_{1,5}^0 \cdot L_{P_5} + l2p_{1,6}^0 \cdot L_{P_6} +}{L_{P_0} + L_{P_1} + L_{P_4} + L_{P_5} + L_{P_6} +} \cdots$$

$$\left.\ldots\frac{l2p^1_{0,0}\cdot L_{P_0}+l2p^1_{0,1}\cdot L_{P_1}+l2p^1_{1,2}\cdot L_{P_2}+l2p^1_{1,3}\cdot L_{P_3}}{L_{P_0}+L_{P_1}+L_{P_2}+L_{P_3}}\right)$$

$$\Leftrightarrow$$

$$\max_{y_k,l2p^k_{ir}}\left(\frac{1\cdot y_0+1\cdot y_1}{1}-\right.$$

$$\left.\frac{l2p^0_{0,0}\cdot 1+l2p^0_{0,1}\cdot 2+l2p^0_{1,5}\cdot 1+l2p^1_{0,0}\cdot 1+l2p^1_{0,1}\cdot 2+l2p^1_{1,2}\cdot 2+l2p^1_{1,3}\cdot 3}{12}\right)$$

with
$L_{P_0}=L_0=1$
$L_{P_1}=L_0+L_2=1+1=2$
$L_{P_2}=L_1=2$
$L_{P_3}=L_1+L_2=2+1=3$
$L_{P_4}=L_3=0$
$L_{P_5}=L_2=1$
$L_{P_6}=L_4=0$

$$\max_{y_k,l2p^k_{ir}}(1.0\cdot y_0+1.0\cdot y_1$$

$$-0.0833\cdot l2p^0_{0,0}-0.1667\cdot l2p^0_{0,1}-0.0833\cdot l2p^0_{1,5}-0\cdot l2p^0_{1,6}-0\cdot l2p^0_{1,4}$$

$$-0.0833\cdot l2p^1_{0,0}-0.1667\cdot l2p^1_{0,1}-0.1667\cdot l2p^1_{1,2}-0.2500\cdot l2p^1_{1,3})$$

All constraints remain unchanged.
The best solution, with regard to latency optimization is:

$y_0=1.0$	$a2c^0_{0,0}=1.0$	$l2p^0_{0,0}=1.0$
$y_1=1.0$	$a2c^0_{1,1}=1.0$	$l2p^0_{1,5}=1.0$
	$a2c^1_{0,0}=1.0$	$l2p^1_{0,0}=1.0$
		$l2p^1_{1,2}=1.0$

All other variables take the value 0. The graphical interpretation of this solution is shown in Figure 4.8.
The latency-aware optimization function tries to map the application nodes as close to the UE group nodes as possible, using the physical path with the smallest latency. Since not all three application nodes can be deployed on c_0, due to its limited CPU capacity, one application node must be mapped on c_1.
It would not be efficient to map a^0_0 on c_1, because then a^0_1 would also have to be mapped on c_1. Otherwise, the latency penalty would be increased even more. Thus, either, a^0_1 or a^1_0 must be allocated on c_1 with its higher latency cost. If a^1_0 is mapped on c_1 then the increased latency caused by using the physical edge e_2 affects both, the virtual link l^1_0 and l^1_1, while allocating a^0_1 on c_1 only increase the latency penalty for l^0_1 by the latency cost of e_2, which is 0.0833 in the objective function for $l2p^0_{1,5}$ compared to a latency cost of 0 for the free-self-path from and to c_0: P_4. The latency penalty when mapping l^1_0 on P_0 and l^1_1 on P_2 is $0.0833+0.1667=0.2500$ instead of

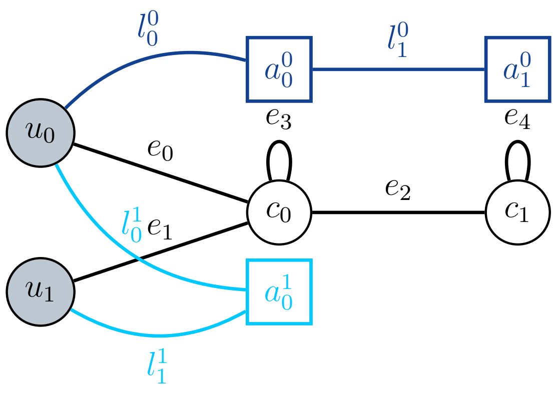


Figure 4.8: Basic Model Example - Latency Optimization

$0.1667 + 0.2500 = 0.4167$ when mapping l_0^1 on P_1 and l_1^1 on P_3. For this reason, a_1^0 is mapped on c_1 while the other two application nodes can be mapped on c_0.

Instead of the latency-aware objective function the cost and revenue objective function can be used. The cost and revenue objective function for the simple NSE example is determined as follows.

$$\max_{y_k, a2c^k_{m,w}, l2p^k_{i,r}} \left(\frac{1 \cdot y_0 + 1 \cdot y_1}{\min\{1,1\}} + \right.$$

$$-\frac{1}{\psi} \cdot \left(a2c^0_{0,0} \cdot \frac{5}{12} + a2c^0_{1,0} \cdot \frac{7}{12} + a2c^1_{0,0} \cdot \frac{4}{12} + a2c^0_{0,1} \cdot \frac{5}{80} + a2c^0_{1,1} \cdot \frac{7}{80} + a2c^1_{0,1} \cdot \frac{4}{80} \right)$$

$$-\frac{1}{\psi} \cdot \left(a2c^0_{0,0} \cdot \frac{7}{10} + a2c^0_{1,0} \cdot \frac{2}{10} + a2c^1_{0,0} \cdot \frac{2}{10} + a2c^0_{0,1} \cdot \frac{7}{100} + a2c^0_{1,1} \cdot \frac{2}{100} + a2c^1_{0,1} \cdot \frac{2}{100} \right)$$

$$-\frac{1}{\psi} \cdot \left(l2p^0_{0,0} \cdot \frac{2}{5} + l2p^0_{0,1} \cdot \left(\frac{2}{5} + \frac{2}{10} \right) + l2p^0_{1,5} \cdot \frac{2}{10} \right)$$

$$\left. -\frac{1}{\psi} \cdot \left(l2p^1_{0,0} \cdot \frac{1}{5} + l2p^1_{0,1} \cdot \left(\frac{1}{5} + \frac{1}{10} \right) + l2p^1_{1,2} \cdot \frac{1}{5} + l2p^1_{1,3} \cdot \left(\frac{1}{5} + \frac{1}{10} \right) \right) \right)$$

with

$$\psi = \frac{5}{12} + \frac{7}{12} + \frac{4}{12} + \frac{5}{80} + \frac{7}{80} + \frac{4}{80} + \frac{7}{10} + \frac{2}{10} + \frac{2}{10} + \frac{7}{100} + \frac{2}{100} + \frac{2}{100}$$
$$+ \frac{2}{5} + \frac{2}{5} + \frac{2}{10} + \frac{2}{10} + \frac{1}{5} + \frac{1}{5} + \frac{1}{10} + \frac{1}{5} + \frac{1}{5} + \frac{1}{10}$$
$$\approx 4.94333$$

$$\max_{y_k, a2c^k_{m,w}, l2p^k_{i,r}} (1 \cdot y_0 + 1 \cdot y_1$$

$$-0.22589 \cdot a2c^0_{0,0} - 0.15846 \cdot a2c^0_{1,0} - 0.10789 \cdot a2c^1_{0,0}$$

$$-0.02680 \cdot a2c^0_{0,1} - 0.02175 \cdot a2c^0_{1,1} - 0.01416 \cdot a2c^1_{0,1}$$

$$-0.08092 \cdot l2p^0_{0,0} - 0.12138 \cdot l2p^0_{0,1} - 0.04046 \cdot l2p^0_{1,5}$$

$$-0.04046 \cdot l2p^1_{0,0} - 0.06069 \cdot l2p^1_{0,1} - 0.04046 \cdot l2p^1_{1,2} - 0.06069 \cdot l2p^1_{1,3})$$

The best solution regarding revenue minus cost is represented by the following variable assignment. The variables not explicitly mentioned here take the value 0.

$y_0 = 1.0$	$a2c^0_{0,0} = 1.0$	$l2p^0_{0,0} = 1.0$
$y_1 = 1.0$	$a2c^0_{1,1} = 1.0$	$l2p^0_{1,5} = 1.0$
	$a2c^1_{0,1} = 1.0$	$l2p^1_{0,1} = 1.0$
		$l2p^1_{1,3} = 1.0$

Figure 4.9 shows a graphical representation of this embedding solution.

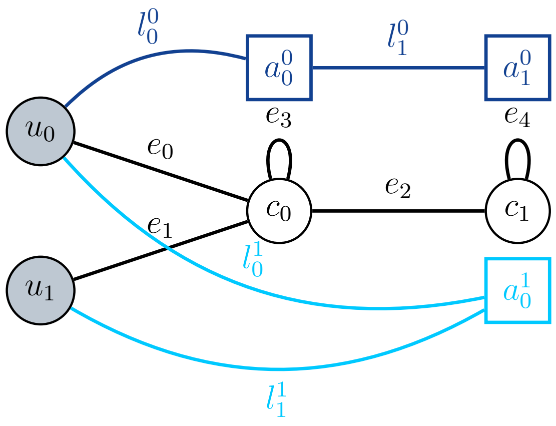


Figure 4.9: Basic Model Example - Cost Optimization

This time, the cost of the used resource, throughput, CPU and memory should be minimized, while embedding as many NSI-Rs as feasible. This leads to the effect, that a^1_0 is allocated on c_1, since the node resources costs predominate the throughput costs here. The node resource cost of mapping a^1_0 on c_0 would have been 0.10789, while they are 0.01416 on c_1 only. Linking a^1_0 when mapped on c_0 has costs of 0.04046 for $l2p^1_{0,0}$ and 0.04046 for $l2p^1_{1,2}$. Compared to that, linking a^1_0, when mapped on c_1 has costs of 0.06069 for $l2p^1_{0,1}$ and 0.06069 for $l2p^1_{1,3}$. Hence, it is cheaper to map a^1_0 on c_1, with total costs of 0.13548, than on c_0, with total costs 0.18881. This does not affect the allocation of the nodes and links from NSI 0, since c_0 and its physical communication links have enough resources to accommodate all applications simultaneously.
For NSI 0, a tradeoff between throughput and node resource costs is made by allocating a^0_0 on c_0 and a^0_1 on c_1. Allocating a^0_0 on c_1 would be cheaper than on c_0, including the throughput costs, but this would violate the latency constraint of l^0_0.

4.3.5.2 Path-Splitting

In this section, the above basic NSE model is enhanced with path-splitting, also referred to as the multi-path approach. That means, the solution considers the possibility to use several paths in the physical network to serve a single link.
The idea of path-splitting is illustrated by the example, presented in Figures 4.10 and 4.11. If a_0^0 is mapped on c_0, then the virtual link l_0^0 can be split among the paths consisting only of e_0 and the path consisting only of e_1. The allocation on e_0 and e_1 must sum up to 1. For instance, 40% of the resources (in our case only throughput) can be provided by e_0 while the remaining 60% of the resources are provided by e_1.

Formalization The model introduced above is very flexible. In order to enable path-splitting only small modifications are needed. First of all, the variable type of the $l2p$ variables has to be modified. Those variables are no longer binary, but continuous values ranging from 0 to 1. I.e., a virtual link l_i^k of a network slice k can be mapped fully, partially or not at all on a physical path P_r:

- if $l2p_{i,r}^k = 1$, then l_i^k is fully mapped on P_r
- if $l2p_{i,r}^k = 0$, then l_i^k is not mapped on P_r
- if $l2p_{i,r}^k = x$ with $x \in (0, 1)$, then P_r provides a share of x of the required capacity of l_i^k

Definition 35 (Continuous Link Mapping Variables)
We define the continuous virtual link to communication path mapping $l2p_{i,r}^k$ for a virtual link l_i^k in the NSIs and a communication path P_r in the substrate as:

$$l2p_{i,r}^k \in [0, 1] \textit{ share of assignment of } l_i^k \textit{ on } P_r \in \mathcal{P}$$

This new definition of the continuous link mapping variables replaces the previous definition of the binary link mapping variables in Definition 13. The small modification of the model has a considerable impact on the solution space, which is largely extended. The acceptance rate for NSI-Rs can be improved, because the path-splitting option can make additional embeddings possible.
Beyond that, the modification of the link to path mapping variables affects the link to edge mapping. While the path to edge mapping in Definition 14 remains unmodified, the link to edge mapping as a product of the new continuously defined link to path mapping and the old path to edge mapping now takes continuous values. The values of the link to edge mappings are to be understood similarly to the values of the $l2p$ variables. Note that the $l2e$ mappings are no new variables, but only a convenient notation for deriving the link to edge mapping from the $l2p$ variables and the fixed $p2e$ parameters.
The other variables (Definition 11 and Definition 12) as well as the parameters of the model remain unchanged.

The latency-aware objective function defined in Definition 17 is a simple mean of achieving that the optimization algorithm prefers mapping the virtual links on substrate links with lower latencies. If path-splitting is activated, the $l2e$ variables are continuous variables with $l2e \in [0,1]$, while they are binary $l2e \in \{0,1\}$ in the single-path scenario. That means, the latencies of the edges of the physical network are weighted with the share of usage of this edge by the links of the accepted NSI-Rs. Therefore, the latency-aware objective function can be used in both scenarios, the path-splitting as well as the single-path scenario. The same applies to the simple as well as the cost and revenue objective function.

In the following the constraints introduced in Section 4.3.5.1 are analyzed towards their transferability to the path-splitting model:

1. Throughput Constraints:
The throughput constraints do not have to be modified, although the $l2e^k_{i,j}$ variables have been changed from binary variables to continuous variables taking values from the interval $[0,1]$.

$$\sum_{\forall k,i} l2e^k_{i,j} \cdot T^k_i \leq T_j, \, \forall j$$

If the resource requirements (in this case the throughput) of the link l^k_i is distributed across several physical paths, i.e., several sets of consecutive edges in the substrate, then $l2e^k_{i,j} \in (0,1)$. Then only the proportionally reserved resources are considered in the constraint. Those are summed up across all links using the respective physical edge.

2. CPU Constraints and 3. Memory Constraints:
The node related constraints, that means, the CPU constraints and the memory constraints are not affected by introducing path splitting. They can be copied from the basic NSE embedding model, see Definitions 22 and 23.

4. Latency Constraints:
The latency constraints of the previous model, from Definition 24

$$l2p^k_{i,r} \cdot L^k_{P_r} \leq L_i, \, \forall k,i,r$$

must be adapted, since the binary $l2p$ mappings result in summing up the full latencies of all physical edges, a specific virtual link has been mapped on. In case of path-splitting, these mappings would only be counted proportionally to the split mappings. This would lead to a proportional consideration of the latency, which would be incorrect. However, this can be fixed, by introducing a correction factor on the right side of the inequation.

Definition 36 (Latency Constraints in Path-Splitting Model)
The latency constraints in the path-splitting NSE model are defined as

$$l2p^k_{i,r} \cdot L_{P_r} \leq l2p^k_{i,r} \cdot L^k_i, \, \forall k,i,r$$

The latency constraints for the path-splitting model state that no virtual link can be mapped on a path in the substrate network which has an overall latency that is

higher than the allowed latency of the virtual link. Definition 36 defines one constraint for each virtual link and physical path combination. Two cases need to be considered when evaluating the inequation in Definition 36.
In the first case, it is assumed that a certain l_i^k is mapped on a path P_r with a rate of $\alpha \in]0, 1)$. The latency L_{P_r} of this path is multiplied by α. However, not the latency weighted by the proportion of link usage should be considered, but rather the sum of latencies of the used edges should be compared with the allowed latency for this link. For instance, if a virtual link l_0^0 is mapped by 40% on a path $P_0 := \{e_0\}$ and by 60% on a path $P_1 := \{e_1\}$, then $l2p_{0,0}^0 = 0.4$ and $l2p_{0,1}^0 = 0.4$. Let's further assume, that the latencies of the edges are: $L_0 = 5$ and $L_1 = 10$ and the allowed latency of the virtual link is $L_0^0 = 8$, then the previous constraints would be $0.4 \cdot 10 \leq 8$ and $0.6 \cdot 5 \leq 8$ This does not reflect the true constraint, which should be $1 \cdot 10 \leq 8$ and $1 \cdot 5 \leq 8$. Thus, we scale the right-hand side of the inequation with $l2p_{i,r}^k = \alpha$.
In the second case, when l_i^k is not mapped on the considered path P_r, the proportion $\alpha = l2p_{i,r}^k$ is zero, therefore the left-hand side of the inequation in Definition 36 is zero, which is always smaller or equal to the right-hand side of the inequation, since the right side cannot become negative.

5. Link Reliability Constraints:
The link reliability constraints which are defined as

$$l2e_{i,j}^k \cdot A_i^k \leq A_j, \quad \forall k, i, j$$

for the basic NSE embedding model in Definition 26 have to be undertaken the same modifications. Like already seen for the latency constraints, the scaling introduced by path splitting on the left-hand side of the inequation must be neutralized by applying the same factor to the right-hand side.

Definition 37 (Link Reliability Constraints in Path-Splitting Model)
The link reliability constraints in the path-splitting NSE model are defined as

$$l2e_{i,j}^k \cdot A_i^k \leq l2e_{i,j}^k \cdot A_j, \quad \forall k, i, j$$

The corresponding $l2p$ mapping link reliability constraints are:

$$l2p_{i,r}^k \cdot A_i^k \leq l2p_{i,r}^k \cdot A_{P_r}, \quad \forall k, i, r$$

6. Network Node Reliability Constraints:
The node reliability constraints are not affected by path-splitting. Definition 28 from the basic NSE model is used.

7. Map Nodes Once Constraints:
The map nodes once constraints are also not affected. Definition 29 is used without modifications.

8. Map Links Once Constraints:
The map links once constraints (see Definition 30) are valid for the continuously

defined $l2p$, variables. The mappings still have to sum up to one, if the NSI-R is embedded, or to zero otherwise.

$$\sum_{\forall r} l2p^k_{i,r} = y_k,\ \forall k, i$$

9. Map Adjacent Links Constraints:
The map adjacent links constraints can be taken over from the simple NSE model, see Definition 31. If a virtual link l^k_i distributes its resource usage across several physical paths, the constraints

$$\sum_{\forall P_r \in \mathcal{P}_{c_v,c_w}} l2p^k_{i,r} = a2c^k_{m,w}\ ,\ \forall k, i, w, m \text{ if } l^k_i = \{a^k_m, a^k_b\}$$

with $\mathcal{P}_{c_v,c_w}$

ensure that the split occupancies of the physical paths sum up to 1, that means, 100% of the virtual link's resource requirements are provided. On the left-hand side, all (partial) resource allocations are summed up. This sum is required to be equal to 1, if the corresponding node mapping of an adjacent application node is applicable.

10. Map UE Links Constraints:
Similarly, the UE links constraints from Definition 32 can be transferred to the path-splitting model.
The map UE links constraints in the basic NSE model are defined as

$$\sum_{\forall P_r \in \mathcal{P}_{u_v,c_w}} l2p^k_{i,r} = y_k\ ,\forall k, i \text{ if } l^k_i = \{u_v, a^k_b\}$$

with $\mathcal{P}_{u_v,c_w} = \mathcal{P}_{c_w,u_v}$

It requires that all virtual links connected with a UE node are mapped on suitable substrate paths, such that the sum of the mapping variables is equal to y_k. Remember that $y_k = 1$ if the k-th NSI is accepted and embedded into the mobile network and $y_k = 0$ otherwise.

11. UE to Application Links Graph Constraints:
The UE to application links graph constraints (see Definition 33) is subjected to a detailed examination. The UE to application links graph constraints in the basic NSE model are defined as

$$a2c^k_{m,w} \geq l2p^k_{i,r},$$
$$\forall k, i, r \text{ with } l^k_i = \{u_v, a^k_m\} \text{ and } P_r \in \mathcal{P}_{c_v,c_w} \text{ with } \mathcal{P}_{c_v,c_w}$$

On the one hand, $a2c$ is a set of binary variables, which can take the values 0 and 1, $l2p$ on the other hand is a set of continuous variables which takes values from the interval $[0, 1]$. I.e., if a link $l^k_i = \{u_v, a^k_m\}$ is fully or partially mapped on a path $P_r \in \mathcal{P}_{c_v,c_w}$ with a share $\alpha \in (0, 1]$, then a^k_m must be mapped on c_w. In other words, if the link to path mapping $l2p^k_{i,r}$ takes a positive value, $a2c^k_{m,w}$ must be set to 1.

12. Map Adjacent Nodes Constraints:
The map adjacent nodes constraints from the basic NSE model in Definition 34 are path-splitting agnostic.

$$a2c^k_{m,v} + a2c^k_{m,w} \geq l2p^k_{i,r},\ \forall k, i, r \text{ with } l^k_i = \{a^k_m, a^k_q\} \text{ and } P_r \in \mathcal{P}_{c_v,c_w}$$
$$a2c^k_{q,v} + a2c^k_{q,w} \geq l2p^k_{i,r},\ \forall k, i, r \text{ with } l^k_i = \{a^k_m, a^k_q\} \text{ and } P_r \in \mathcal{P}_{c_v,c_w}$$
$$a2c^k_{q,w} \geq l2p^k_{i,r},\ \forall k, i, r \text{ with } l^k_i = \{u_v, a^k_q\} \text{ and } P_r \in \mathcal{P}_{u_v,c_w}$$
$$a2c^k_{m,v} \geq l2p^k_{i,r},\ \forall k, i, r \text{ with } l^k_i = \{a^k_m, a^k_q\} \text{ and } P_r = \mathcal{P}^{free}_{c_v}$$
$$a2c^k_{q,v} \geq l2p^k_{i,r},\ \forall k, i, r \text{ with } l^k_i = \{a^k_m, a^k_q\} \text{ and } P_r = \mathcal{P}^{free}_{c_v}$$

Changing the formerly binary $l2p$ variables, to continuous variables with a value range from 0 to 1 does not alter the validity of the constraints. Once the $l2p$ mapping on the right-hand side of one of the inequation is greater than zero the corresponding application mappings are triggered. Yet, it is irrelevant whether $l2p$ takes the value 1 as the only option greater than zero in the basic NSE model or any other positive value in the path-splitting NSE model.

Summary of the Path-Splitting NSE Model In the following, the condensed path-splitting model considering the modifications proposed above is provided. The modifications are highlighted in blue color.
Note that the latency-aware objective function is interchangeable with the objective functions introduced in Section 4.3.4.

Model 2 (Path-Splitting NSE Model)

$$\max_{y_k, l2p^k_{i,r}} \left(\left(\sum_{\forall k} y_k \frac{\omega_k}{\min_k \omega_k} \right) - \left(\frac{\sum_{\forall k,i,r} l2p^k_{i,r} \cdot L_{P_r}}{\sum_{\forall k,i,r} l2p'^k_{i,r} \cdot L_{P_r}} \right) \right)$$

under

1. *Throughput Constraints:*

$$\sum_{\forall k,i} l2e^k_{i,j} \cdot T^k_i \leq T_j,\ \forall j$$

2. *CPU Constraints:*

$$\sum_{\forall k,m} a2c^k_{m,w} \cdot D^k_m \leq D_w,\ \forall w$$

3. *Memory Constraints:*

$$\sum_{\forall k,m} a2c^k_{m,w} \cdot M^k_m \leq M_w,\ \forall w$$

4. *Latency Constraints:*

$$l2p_{i,r}^{k} \cdot L_{P_r} \leq l2p_{i,r}^{k} \cdot L_i^k, \; \forall k, i, r$$

5. *Link Reliability Constraints:*

$$l2e_{i,j}^{k} \cdot A_i^k \leq l2e_{i,j}^{k} \cdot A_j, \quad \forall k, i, j$$

6. *Network Node Reliability Constraints:*

$$a2c_{m,w}^{k} \cdot B_m^k \leq B_w, \quad \forall k, m, w$$

7. *Map Nodes Once Constraints:*

$$\sum_{\forall w} a2c_{m,w}^{k} = y_k, \; \forall k, m$$

8. *Map Links Once Constraints:*

$$\sum_{\forall r} l2p_{i,r}^{k} = y_k, \; \forall k, i$$

9. *Map Adjacent Links Constraints:*

$$\sum_{\forall P_r \in \mathcal{P}_{c_v,c_w}} l2p_{i,r}^{k} \geq a2c_{m,w}^{k} \; ,\forall k, i, w, m \; \textit{if} \; l_i^k = \{a_m^k, a_b^k\} \; \textit{or} \; l_i^k = \{a_b^k, a_m^k\}$$

10. *Map UE Links Constraints:*

$$\sum_{\forall P_r \in \mathcal{P}_{u_v,c_w}} l2p_{i,r}^{k} = y_k \; ,\forall k, i, \; \textit{if} \; l_i^k = \{u_v, a_b^k\}$$

11. *UE to Application Links Graph Constraints:*

$$a2c_{m,w}^{k} \geq l2p_{i,r}^{k}, \; \forall k, i, r \; \textit{with} \; l_i^k = \{u_v, a_m^k\} \; \textit{and} \; P_r \in \mathcal{P}_{c_v,c_w}$$

12. *Map Adjacent Nodes Constraints:*
    a) *for Cloud Node to Cloud Node Paths:*

$$a2c_{m,v}^{k} + a2c_{m,w}^{k} \geq l2p_{i,r}^{k}, \; \forall k, i, r \; \textit{with} \; l_i^k = \{a_m^k, a_q^k\} \; \textit{and} \; P_r \in \mathcal{P}_{c_v,c_w} \; \textit{and}$$

$$a2c_{q,v}^{k} + a2c_{q,w}^{k} \geq l2p_{i,r}^{k}, \; \forall k, i, r \; \textit{with} \; l_i^k = \{a_m^k, a_q^k\} \; \textit{and} \; P_r \in \mathcal{P}_{c_v,c_w}$$

    b) *for UE Group Node to Cloud Node Paths:*

$$a2c_{q,w}^{k} \geq l2p_{i,r}^{k}, \; \forall k, i, r \; \textit{with} \; l_i^k = \{u_v, a_q^k\} \; \textit{and} \; P_r \in \mathcal{P}_{u_v,c_w}$$

    c) *for Free-Self Paths:*

$$a2c_{m,v}^{k} \geq l2p_{i,r}^{k}, \; \forall k, i, r \; \textit{with} \; l_i^k = \{a_m^k, a_q^k\} \; \textit{and} \; P_r = \mathcal{P}_{c_v}^{free} \; \textit{and}$$

$$a2c_{q,v}^{k} \geq l2p_{i,r}^{k}, \; \forall k, i, r \; \textit{with} \; l_i^k = \{a_m^k, a_q^k\} \; \textit{and} \; P_r = \mathcal{P}_{c_v}^{free}$$

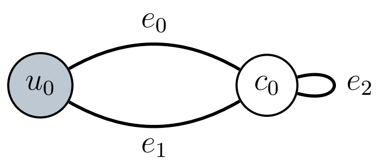


Figure 4.10: Path-Splitting Example - Substrate Network

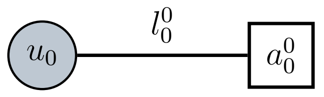


Figure 4.11: Path-Splitting Example - Network Slice 0

Table 4.8: Path-Splitting Example - Network Node Parameters

Node	**CPU**	**Memory**	**Reliability**
c_0	20	10	1.00
a_0^0	18	9	0.90

Table 4.9: Path-Splitting Example - Network Link Parameters

Link	**Throughput**	**Latency**	**Reliability**
e_0	20	1	1.00
e_1	20	2	1.00
e_2	∞	0	1.00
l_0^0	25	5	0.90

Example The following basic example is used to demonstrate the concept and solution of the path-splitting NSE model.
The substrate in Figure 4.10 comprises only one cloud node which is connected to one UE group with two alternative physical links.
Only one NSI-R with weight $\omega_0 := 1$ is defined for this minimal path-splitting example, see Figure 4.11.
The associated resources and capabilities are shown in the Tables 4.8 and 4.9.
When formalizing this small path-splitting example as an MILP as explained above, the following physical paths and variables are defined.

$$P_0 := \{e_1\}$$
$$P_1 := \{e_0\}$$
$$P_2 := P_{c_0}^{free} = \{e_2\}$$

The binary mapping variables y_0 and $a_{0,0}^0$ are defined. Beyond that, the continuous $l2p$ mapping variables $l_{0,0}^0$ and $l_{0,1}^0$ are defined. Note that, $l_{0,2}^0$ is not a variable, since e_2 is not adjacent to u_0.

The latency-aware objective function looks as follows for this example:

$$\max_{y_k, l2p_{i,r}^k} \left(y_0 - \frac{2 \cdot l2p_{0,0}^0 + 1 \cdot l2p_{0,1}^0}{2+1} \right)$$
$$\Leftrightarrow$$
$$\max_{y_k, l2p_{i,r}^k} \left(y_0 - \frac{2}{3} \cdot l2p_{0,0}^0 - \frac{1}{3} \cdot l2p_{0,1}^0 \right)$$

The most insightful constraints in this example are the throughput constraints. They are defined as:

Throughput constraints for e_0: $25 \cdot l2p_{0,0}^0 \leq 20$

Throughput constraints for e_1: $25 \cdot l2p_{0,1}^0 \leq 20$

Both throughput constraints have in common, that they can only be fulfilled when the value of the respective $l2p$ variable is smaller or equal 0.8. Since the latency cost of $l2p_{0,0}^0$ is higher than the latency cost of $l2p_{0,1}^0$ we get the following result when solving the MILP with an appropriate out-of-the-box solver.

$$y_0 = 1 \qquad a2c_{0,0}^0 = 1 \qquad l2p_{0,0}^0 = 0.2$$
$$l2p_{0,1}^0 = 0.8$$

That means, the virtual link l_0^0 is split on the two available physical paths P_0 and P_1. P_1 provides 80% of the required throughput resources, while P_0 provides the remaining 20%.

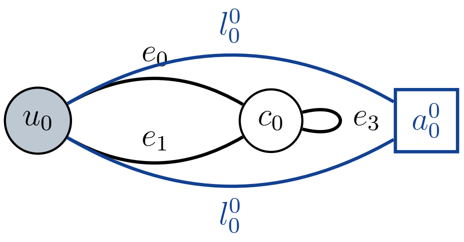


Figure 4.12: Path-Splitting Example - Latency-Aware Optimization

4.3.5.3 Multiple Application Instantiation NSE Model

Edge computing is seen as a key feature of 5G to enable local communication, with a very low latency and high throughput, while at the same time improving data security and reducing transmission costs. It provides computation power and data storage close to the mobile users, within the 5G RAN, at the so-called edge of the mobile network. Edge clouds are typically deployed on 5G gNodeBs, i.e., macro base stations or the multi-Radio Access Technology cell aggregation sites, instead of centralized core nodes. This leverages location-based services, like the communication between co-located devices in car-to-x and smart factory applications as well as, for instance, augmented reality services. By using edge computing the latency and volume of the transferred data is reduced. Hence, the available bandwidth can be used more efficiently. Additionally, edge clouds can be used for computation offloading, enabling services that cannot be performed by most mobile device, e.g., Artificial Intelligence (AI) based services. Furthermore, a better QoE and a lower battery consumption of the UEs can be provided by edge computing. However, edge clouds come at high Operating Expenditures (OPEX). Therefore, edge cloud resources are limited, while their applications are constantly increasing.[52]

Allocating edge cloud resources is not considered in conventional VNE solutions. Deploying a service close to its users on an edge cloud often means that the same service has to be provided at different locations, i.e., for one application in an NSI several instances might have to be deployed in the substrate network. In this section, a formal, mathematical model of the NSE problem leveraging edge computing is provided.

Formalization The MAI NSE model does not necessitate any modifications of the variables and parameters. However, the MAI NSE model requires, that an application a_m^k from the k-th network slice can be embedded several times on different cloud nodes in the substrate network. If the k-th network slice is accepted, every application has to be instantiated at least once. In the Definition of the basic NSE model (see Section 4.3.5.1) every application of an accepted network slice is mapped exactly once, as defined in Definition 29:

$$\sum_{\forall w} a2c^k_{m,w} = y_k, \, \forall k, m$$

In order to enable MAI, it is changed to the following definition.

Definition 38 (Map Nodes Constraints in MAI NSE Model)
The map nodes constraints in the MAI NSE model are defined as

$$\sum_{\forall w} a2c^k_{m,w} \geq y_k,\ \forall k, m$$

In the previous models, every virtual link is mapped in the substrate network exactly once, if the network slice is embedded. However, in the MAI NSE model, several instances of one virtual application node might have been created. Those have to be connected with their adjacent links. Thus, several instances of the same virtual link might be needed in the mapping. Therefore, Definition 30

$$\sum_{\forall r} l2p^k_{i,r} = y_k,\ \forall k, i$$

is modified as follows.

Definition 39 (Map Links Constraints in MAI NSE Model)
The map links constraints in the MAI NSE model are defined as

$$\sum_{\forall r} l2p^k_{i,r} \geq y_k,\ \forall k, i$$

The same applies to the map adjacent links constraints as well as the map UE links constraints. Since in the MAI NSE model several instances of the same application might have been created for different UE groups, those have to be connected with their adjacent link mappings. The map adjacent links constraints from the basic NSE model (see Definition 31) as well as the map UE links constraints (see Definition 32) do not have to be modified.

$$\sum_{\forall P_r \in \mathcal{P}_{c_v,c_w}} l2p^k_{i,r} \geq a2c^k_{m,w}\ , \forall k, i, w, m \text{ if } l^k_i = \{a^k_m, a^k_b\}$$

and

$$\sum_{\forall P_r \in \mathcal{P}_{u_v,c_w}} l2p^k_{i,r} = y_k\ , \forall k, i \text{ if } l^k_i = \{u_v, a^k_b\}$$

If two applications are connected and one of these two or both are instantiated multiple times, then the adjacent links, must be mapped multiple times. Thus, the number of $l2p$ mappings must be greater or equal than the number of $a2c$ mappings. However, each UE group is connected to exactly one instance of each adjacent application. Since a UE does not need to be connected to more than one instance of a directly connected application, the map UE links constraints from Definition 32 remains unchanged.

The UE to application links graph constraints, specified in Definition 33, are not affected by the MAI model. The constraints

$$a2c^k_{m,w} \geq l2p^k_{i,r},\ \forall k, i, r \text{ with } l^k_i = \{u_v, a^k_m\} \text{ and } P_r \in \mathcal{P}_{c_v,c_w}$$

are still valid if the respective application a_m^k is instantiated on several cloud nodes.

Moreover, the map adjacent nodes constraints specified in Definition 34 have been created to be applicable to the basic, path-splitting as well as the MAI model. They ensure a valid combination of the link and path mappings in the solution, such that the embedded nodes are physically connected as defined in the NSI-R. Independent of the number of instances created for an application in the substrate network the adjacent physical links must be embedded on suitable physical communication paths. The remaining constraints defined by the basic NSE model in Section 4.3.5.1, namely the throughput constraints, the CPU constraints, the memory constraints, the latency constraints, the link reliability constraints and the network node reliability constraints remain unchanged.

Summary of the MAI NSE Model This results in the following full MAI NSE Model.

Model 3 (MAI NSE Model)

$$\max_{y_k, a2c_{m,w}^k, l2p_{i,r}^k} \left(\left(\sum_{\forall k} y_k \frac{\omega_k}{\min\limits_k \omega_k} \right) - \frac{\sum_{\forall k,m,w} a2c_{m,w}^k \cdot \frac{D_m^k}{D_w} + \sum_{\forall k,m,w} a2c_{m,w}^k \cdot \frac{M_m^k}{M_w} + \sum_{\forall k,i,j} l2e_{i,j}^k \cdot \frac{T_i^k}{T_j}}{\sum_{\forall k,m,w} \frac{D_m^k}{D_w} + \sum_{\forall k,m,w} \frac{M_m^k}{M_w} + \sum_{\forall k,i,j} \frac{T_i^k}{T_j}} \right)$$

under

1. *Throughput Constraints:*

$$\sum_{\forall k,i} l2e_{i,j}^k \cdot T_i^k \leq T_j, \; \forall j$$

2. *CPU Constraints:*

$$\sum_{\forall k,m} a2c_{m,w}^k \cdot D_m^k \leq D_w, \; \forall w$$

3. *Memory Constraints:*

$$\sum_{\forall k,m} a2c_{m,w}^k \cdot M_m^k \leq M_w, \; \forall w$$

4. *Latency Constraints:*

$$l2p_{i,r}^k \cdot L_{P_r} \leq L_i^k, \; \forall k, i, r$$

5. *Link Reliability Constraints:*

$$l2e_{i,j}^k \cdot A_i^k \leq A_j, \quad \forall k, i, j$$

6. *Network Node Reliability Constraints:*
$$a2c^k_{m,w} \cdot B^k_m \leq B_w, \quad \forall k, m, w$$
7. *Map Nodes Constraints:*
$$\sum_{\forall w} a2c^k_{m,w} \geq y_k, \forall k, m$$
8. *Map Links Constraints:*
$$\sum_{\forall r} l2p^k_{i,r} \geq y_k, \forall k, i$$
9. *Map Adjacent Links Constraints:*
$$\sum_{\forall P_r \in \mathcal{P}_{c_v,c_w}} l2p^k_{i,r} \geq a2c^k_{m,w} \text{ ,} \forall k, i, w, m \textit{ if } l^k_i = \{a^k_m, a^k_b\}$$
10. *Map UE Links Constraints:*
$$\sum_{\forall P_r \in \mathcal{P}_{u_v,c_w}} l2p^k_{i,r} = y_k \text{ ,} \forall k, i, \textit{ if } l^k_i = \{u_v, a^k_b\}$$
11. *UE to Application Links Graph Constraints:*
$$a2c^k_{m,w} \geq l2p^k_{i,r}, \forall k, i, r \textit{ with } l^k_i = \{u_v, a^k_m\} \textit{ and } P_r \in \mathcal{P}_{c_v,c_w}$$
12. *Map Adjacent Nodes Constraints:*
    a) *for Cloud Node to Cloud Node Paths:*
$$a2c^k_{m,v} + a2c^k_{m,w} \geq l2p^k_{i,r}, \forall k, i, r \textit{ with } l^k_i = \{a^k_m, a^k_q\} \textit{ and } P_r \in \mathcal{P}_{c_v,c_w} \textit{ and}$$
$$a2c^k_{q,v} + a2c^k_{q,w} \geq l2p^k_{i,r}, \forall k, i, r \textit{ with } l^k_i = \{a^k_m, a^k_q\} \textit{ and } P_r \in \mathcal{P}_{c_v,c_w}$$
    b) *for UE Group Node to Cloud Node Paths:*
$$a2c^k_{q,w} \geq l2p^k_{ir}, \forall k, i, r \textit{ with } l^k_i = \{u_v, a^k_q\} \textit{ and } P_r \in \mathcal{P}_{u_v,c_w}$$
    c) *for Free-Self-Paths:*
$$a2c^k_{m,v} \geq l2p^k_{i,r}, \forall k, i, r \textit{ with } l^k_i = \{a^k_m, a^k_q\} \textit{ and } P_r = \mathcal{P}^{free}_{c_v} \textit{ and}$$
$$a2c^k_{q,v} \geq l2p^k_{i,r}, \forall k, i, r \textit{ with } l^k_i = \{a^k_m, a^k_q\} \textit{ and } P_r = \mathcal{P}^{free}_{c_v}$$

Implementation Details The MAI model allows to create several instances of the same application. If the simple or latency-aware objective function is used, the embedding result might be inefficient regarding the number of application instances, since node resource minimization is not part of those objective functions. Therefore, the cost and revenue objective function should be used.
In addition, dispensable mappings on free-self paths might occur. This is resolved by the following implementation: When using the cost and revenue objective function, one wants an embedding of the network slices which uses the minimum amount of resources and is free for dispensable mappings, especially unnecessary free-self path mappings. In order to achieve this, in this implementation the link to free-self path mapping variables are assigned with a tiny cost of $1 \cdot 10^{-1}$. This prevents dispensable free-self path mappings without modifying the NSE otherwise.

Example This example illustrates the characteristics and solution of the MAI NSE model. The substrate and one NSI-R are shown in the Figures 4.13 and 4.14.

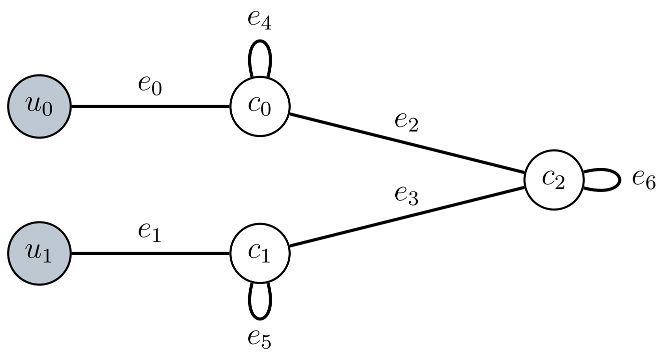


Figure 4.13: MAI Example - Substrate Network

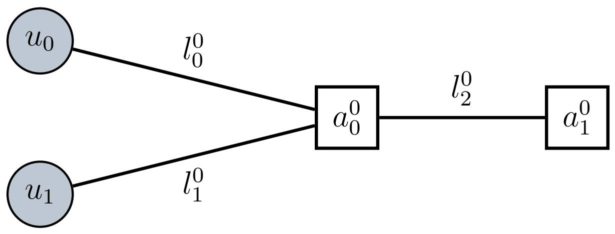


Figure 4.14: MAI Example - Network Slice 0

The provided resources and capabilities of the substrate as well as the required resources and capabilities of the NSI-R are compiled in the Tables 4.10 and 4.11.

Table 4.10: MAI Example - Network Node Parameters

Node	**CPU**	**Memory**	**Reliability**
c_0	10	10	0.93
c_1	10	10	0.93
c_2	10	10	0.93
a_0^0	10	10	0.90
a_1^0	10	10	0.90

Table 4.11: MAI Example - Network Link Parameters

Link	**Throughput**	**Latency**	**Reliability**
e_0	100	1	0.80
e_1	100	1	0.80
e_2	100	1	0.80
e_3	100	1	0.80
e_4	∞	0	1.00
e_5	∞	0	1.00
e_6	∞	0	1.00
l_0^0	100	1.5	0.80
l_1^0	100	1.5	0.80
l_2^0	100	1.5	0.80

The physical paths of the substrate in this example are defined as follows:

$$
\begin{aligned}
&P_0 := \{e_0\}\\
&P_1 := \{e_0, e_2\}\\
&P_2 := \{e_0, e_2, e_3\}\\
&P_3 := \{e_1\}\\
&P_4 := \{e_1, e_3\}\\
&P_5 := \{e_1, e_3, e_2\}\\
&P_6 := P_{c0}^{free} = \{e_4\}\\
&P_7 := \{e_2\}\\
&P_8 := \{e_2, e_3\}\\
&P_9 := P_{c_1}^{free} = \{e_5\}\\
&P_{10} := \{e_3\}\\
&P_{11} := P_{c_2}^{free} = \{e_6\}
\end{aligned}
$$

The ILP model for solving the NSE problem with MAI defines the binary variable y_0 as well as the binary $a2c$ variables shown in Table 4.12 and the $l2p$ variables shown in Table 4.13. The binary $l2p$ variables are based on the feasible combinations of virtual link to physical path mappings.

Table 4.12: MAI Example - Application to Cloud Node Mapping Variables

	a_0^0	a_1^0
c_0	$a2c_{0,0}^0$	$a2c_{1,0}^0$
c_1	$a2c_{0,1}^0$	$a2c_{1,1}^0$
c_2	$a2c_{0,2}^0$	$a2c_{1,2}^0$

Table 4.13: MAI Example - Link to Path Mapping Variables

	l_0^0	l_1^0	l_2^0
P_0	$l2p_{0,0}^0$	-	-
P_1	$l2p_{0,1}^0$	-	-
P_2	$l2p_{0,2}^0$	-	-
P_3	-	$l2p_{1,3}^0$	-
P_4	-	$l2p_{1,4}^0$	-
P_5	-	$l2p_{1,5}^0$	-
P_6	-	-	$l2p_{2,6}^0$
P_7	-	-	$l2p_{2,7}^0$
P_8	-	-	$l2p_{2,8}^0$
P_9	-	-	$l2p_{2,9}^0$
P_{10}	-	-	$l2p_{2,10}^0$
P_{11}	-	-	$l2p_{2,11}^0$

There is no possibility to embed the application a_0^0 into the substrate network such that both UE groups can reach it with the required latency. If a_0^0 is embedded on c_0 the required latency of the connection between u_1 and a_0^0 cannot be fulfilled. If it is mapped on c_1 the required latency for the connection with u_0 is violated and when it is mapped on c_2 the latency requirements of both UE groups cannot be fulfilled. To solve this problem, two instances of a_0^0 are created in the MAI NSE solution. The first instance of a_0^0 is mapped on c_0 to serve u_0 and the second instance is mapped on c_1 to serve u_1.

The full solution is provided by the following variable assignment. All variables which are not mentioned here have the value zero.

$$
\begin{array}{lll}
y_0 = 1 & a2c_{0,0}^0 = 1 & l2p_{0,0}^0 = 1 \\
 & a2c_{0,1}^0 = 1 & l2p_{1,3}^0 = 1 \\
 & a2c_{1,2}^0 = 1 & l2p_{2,7}^0 = 1 \\
 & & l2p_{2,10}^0 = 1
\end{array}
$$

Figure 4.15 illustrates the embedding solution with the two instances of the application a_0^0.

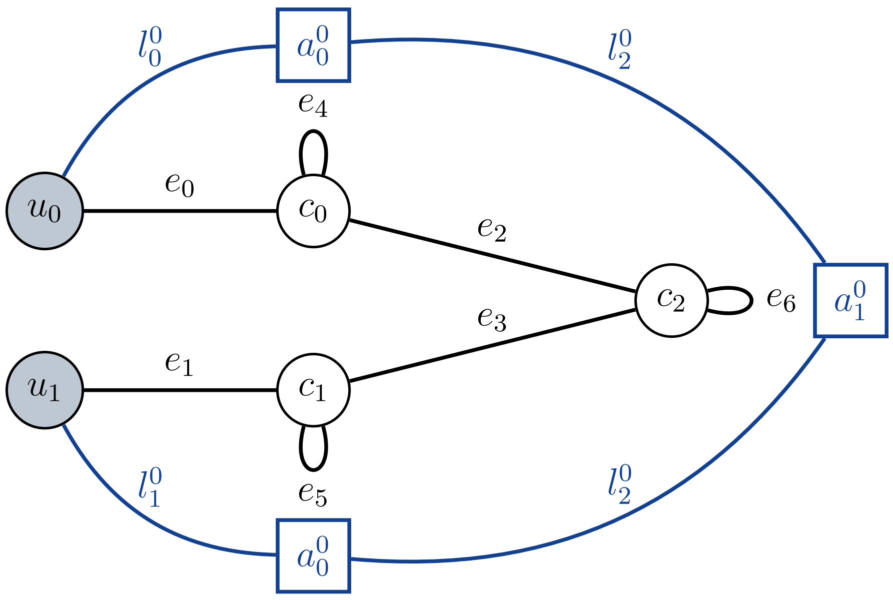


Figure 4.15: MAI Example - Cost Optimization

4.3.5.4 Combined Path-Splitting and Multiple Application Instantiation NSE Model

The two variations of the NSE model introduced above providing the capabilities of path-splitting and MAI can be combined. This way, path-splitting as well as MAI can be enabled concurrently.
In this implementation the cost and revenue objective function is used, since it avoids unnecessary free-self-path mappings when MAI is active.

Model 4 (Combined Path-Splitting and MAI NSE Model)

$$\max_{y_k, a2c^k_{m,w}, l2p^k_{i,r}} \left(\left(\sum_{\forall k} y_k \frac{\omega_k}{\min\limits_k \omega_k} \right) - \right.$$

$$\left. \frac{\sum_{\forall k,m,w} a2c^k_{m,w} \cdot \frac{D^k_m}{D_w} + \sum_{\forall k,m,w} a2c^k_{m,w} \cdot \frac{M^k_m}{M_w} + \sum_{\forall k,i,j} l2e^k_{i,j} \cdot \frac{T^k_i}{T_j}}{\sum_{\forall k,m,w} \frac{D^k_m}{D_w} + \sum_{\forall k,m,w} \frac{M^k_m}{M_w} + \sum_{\forall k,i,j} \frac{T^k_i}{T_j}} \right)$$

under

1. *Throughput Constraints:*

$$\sum_{\forall k,i} l2e^k_{i,j} \cdot T^k_i \leq T_j, \, \forall j$$

2. *CPU Constraints:*

$$\sum_{\forall k,m} a2c^k_{m,w} \cdot D^k_m \leq D_w, \, \forall w$$

3. *Memory Constraints:*

$$\sum_{\forall k,m} a2c^k_{m,w} \cdot M^k_m \leq M_w, \, \forall w$$

4. *Latency Constraints:*

$$l2p^k_{i,r} \cdot L_{P_r} \leq l2p^k_{i,r} \cdot L^k_i, \, \forall k, i, r$$

5. *Link Reliability Constraints:*

$$l2e^k_{i,j} \cdot A^k_i \leq l2e^k_{i,j} \cdot A_j, \quad \forall k, i, j$$

6. *Network Node Reliability Constraints:*

$$a2c^k_{m,w} \cdot B^k_m \leq B_w, \quad \forall k, m, w$$

7. *Map Nodes Constraints:*

$$\sum_{\forall w} a2c^k_{m,w} \geq y_k, \, \forall k, m$$

8. *Map Links Constraints:*

$$\sum_{\forall r} l2p^k_{i,r} \geq y_k, \, \forall k, i$$

9. *Map Adjacent Links Constraints:*

$$\sum_{\forall P_r \in \mathcal{P}_{c_v,c_w}} l2p^k_{i,r} \geq a2c^k_{m,w} \;, \forall k, i, w, m \; \textit{if} \; l^k_i = \{a^k_m, a^k_b\}$$

10. *Map UE Links Constraints:*

$$\sum_{\forall P_r \in \mathcal{P}_{u_v,c_w}} l2p^k_{i,r} = y_k \;, \forall k, i, \; \textit{if} \; l^k_i = \{u_v, a^k_b\}$$

11. *UE to Application Links Graph Constraints:*

$$a2c^k_{m,w} \geq l2p^k_{i,r}, \, \forall k, i, r \; \textit{with} \; l^k_i = \{u_v, a^k_m\} \; \textit{and} \; P_r \in \mathcal{P}_{c_v,c_w}$$

12. *Map Adjacent Nodes Constraints:*
    a) *for Cloud Node to Cloud Node Paths:*

$$a2c^k_{m,v} + a2c^k_{m,w} \geq l2p^k_{i,r}, \, \forall k, i, r \; \textit{with} \; l^k_i = \{a^k_m, a^k_q\} \; \textit{and} \; P_r \in \mathcal{P}_{c_v,c_w} \; \textit{and}$$

$$a2c^k_{q,v} + a2c^k_{q,w} \geq l2p^k_{i,r}, \, \forall k, i, r \; \textit{with} \; l^k_i = \{a^k_m, a^k_q\} \; \textit{and} \; P_r \in \mathcal{P}_{c_v,c_w}$$

    b) *for UE Group Node to Cloud Node Paths:*

$$a2c^k_{q,w} \geq l2p^k_{i,r}, \, \forall k, i, r \; \textit{with} \; l^k_i = \{u_v, a^k_q\} \; \textit{and} \; P_r \in \mathcal{P}_{u_v,c_w}$$

    c) *for Free-Self-Paths:*

$$a2c^k_{m,v} \geq l2p^k_{i,r}, \, \forall k, i, r \; \textit{with} \; l^k_i = \{a^k_m, a^k_q\} \; \textit{and} \; P_r = \mathcal{P}^{free}_{c_v} \; \textit{and}$$

$$a2c^k_{q,v} \geq l2p^k_{i,r}, \, \forall k, i, r \; \textit{with} \; l^k_i = \{a^k_m, a^k_q\} \; \textit{and} \; P_r = \mathcal{P}^{free}_{c_v}$$

5 Probabilistic Network Slice Embedding

This chapter analyzes the probabilistic NSE, also referred to as NSE under uncertainty. The main assumption made in Chapter 4 is that the resources of the substrate network are constantly provided, while there is also no fluctuation in the resource demands of the NSIs. In this section, a more realistic model is presented, analyzing the resource allocation problem under uncertain resources and demands. The so-called probabilistic NSE model is based on the optimal NSE model variants specified in Chapter 4. All model variants can serve as a basis for the probabilistic NSE model. A new uncertainty-aware objective function is defined for the probabilistic NSE model. Figure 5.1 provides an overview over the optimal and probabilistic NSE model variants.

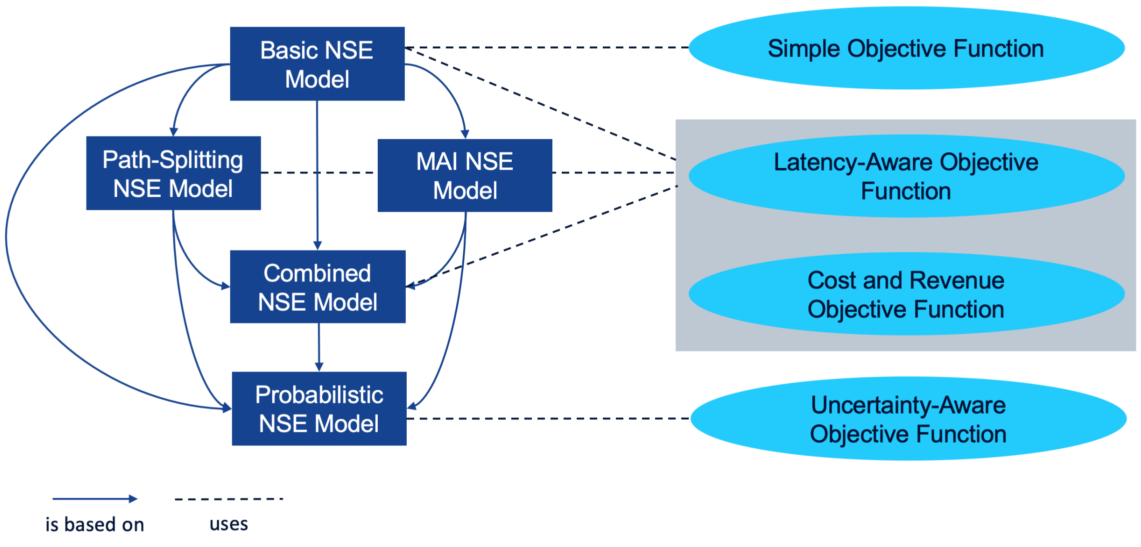

Basic NSE Model
Path-Splitting NSE Model
MAI NSE Model
Combined NSE Model
Probabilistic NSE Model
Simple Objective Function
Latency-Aware Objective Function
Cost and Revenue Objective Function
Uncertainty-Aware Objective Function
is based on
uses

Figure 5.1: Big Picture of the Optimal and Probabilistic NSE Model Variants

In this chapter, first of all the problem and motivation as well as the goals and requirements of NSE under uncertainty are described. Then the probabilistic NSE model is defined formally. Moreover, important uncertainty metrics are analyzed. This includes the so-called Probability of Feasibility (POF) and the safety buffers. Finally, the approach is illustrated by a comprehensible example.

5.1 Problem and Motivation

The fifth generation (5G) of mobile networks support several new use cases, like the IoT, mMTC and URLLC as well as significant improvements of the conventional Mobile Broadband (MBB) use case. The virtual separation of network slices on a common end-to-end mobile network infrastructure enables an efficient usage of the underlying network resources and provides means for security and safety related isolation of the defined logical networks. A much-discussed challenge is the overbooking of resources guaranteed by contract.
For an efficient and beneficial spectrum use, especially in the RAN, moderate physical network resource overbooking is indispensable, since the peak NSI resource requirements are unlikely to be requested by all NSIs simultaneously in most use cases. Thus, for scarce resources, like mobile network frequency bands, the expected rather than the worst-case network resource availability should be assumed.
Careful resource overbooking is acceptable for the majority of the 5G mobile network use cases and it is unavoidable for efficient and fair resource utilization in future 5G mobile networks.[4]

5.2 Goals and Requirements

In this chapter, an efficient model for probabilistic NSE is presented which enables an informed decision on NSIA under uncertainty. It is based on the guaranteed end-to-end mobile network resources that have to be provided to the NSIs on the one hand and the uncertain capacities and capabilities of the underlying network infrastructure on the other hand.

A two-step approach is proposed. In the first step, the best NSE with respect to robust resource provisioning is determined. In the second step, the degree of robustness, or the risk of SLA violation for an embedded NSI (i.e., the probability of failure to provide the guaranteed resources) is analyzed. If the decision-maker considers the robustness as good enough, i.e., if the probability that the actual required resources will be available to the NSI when requested is acceptable, then the NSIs can be deployed according to the embedding determined in the first step. This might imply an overbooking of the physical network resources.

In order to maintain an LP that can be solved efficiently, the uncertainty in the resource availability and utilization are addressed in the objective function only. The linear constraints of the NSE optimization problem use the expected values (means) for the resource availability and utilization. The objective function ensures that as many NSIs are embedded as possible, the allocation minimizes uncertainty and the most beneficial NSIs are selected if the network infrastructure does not provide enough resources.

5.3 Problem Formalization

This model is based on the previous model presented in Chapter 4, which assumes full and exact knowledge on the available resources. End-to-end mobile network resources, like the throughput of the communication links as well as the computation power and memory on the cloud servers, underlie fluctuations and cannot be predicted accurately.
Therefore, the model proposed in Chapter 4 is enhanced to reflect these uncertainties and find robust NSE solutions. For robust NSE the expected resource demands, plus a safety buffer and the expected available resources minus a safety buffer are used. The uncertainty-aware objective function guarantees a beneficial NSI embedding while controlling uncertainties and potential overbookings.

5.3.1 Parameter

The notation introduced in Chapter 4 is used for the NSE under uncertainty model. The uncertainty of the resource parameters is modeled with an arbitrary stochastic distribution function.

In this thesis, normal distributions are used to describe uncertain demands. $\mathcal{N}(\mu, \sigma)$ denotes a normal distribution with the mean μ and the Standard Deviation (STD) σ. The normal distribution is also referred to as Gaussian distribution. $\mathcal{N}(0, 1)$ is the so-called standard-normal distribution. The probability density function of the normal distribution is defined as:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Figure 5.2 shows the probability density function of the Gaussian distribution.

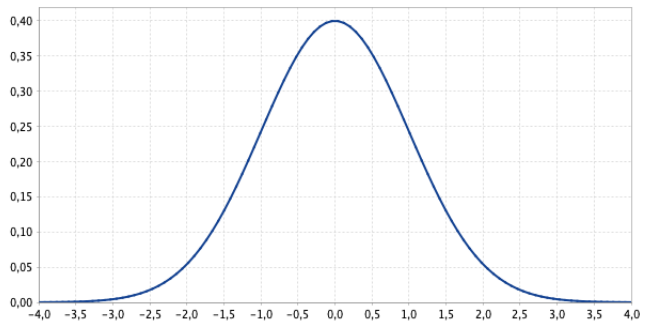


Figure 5.2: Normal Distribution Density Function

The cumulative distribution function of the standard normal distribution is

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}t^2} dt$$

For other μ and σ the cumulative normal distribution is

$$F(x) = \Phi(\frac{x-\mu}{\sigma})$$

Using Gaussian distributions for the resource availability and demands is a simplification which assumes that, there is an expected value for the amount of a provided or requested resource and values close to this mean are realized with a higher probability, than values further away from the mean value. Each resource on every network elements has its specific mean and STD.

Note that the Robust NSE Model is applicable for any arbitrary stochastic distribution function, to be used for the uncertain resources and demands. Solving the NSE problem as described in the following is independent of any specific distribution functions underlying its parameter. Naturally, the calculation of the POFs are based on the concrete distribution function.

Every resource, capability, demand or requirement within the NSE model can underlie uncertainty. Throughout this thesis, we assume, that the provided node resources and capabilities in general do not underlie uncertainty. Therefore, we model them similarly to Section 4.3.2 as O_w, D_w, M_w, W_w and B_w, as summarized in Table 5.1 for the node resource. The resource requirements of the virtual nodes in the NSIs are uncertain. Therefore, their mean and STD are defined in Table 5.1.

Table 5.1: Certain and Uncertain Node Resources and Capabilities

Def.	**Type**	**Description**
O_w	GCR	Provided resource of substrate node c_w
D_w	SCR	Provided computation capacity of substrate node c_w
M_w	SCR	Provided memory capacity of substrate node c_w
W_w	GCQ	Provided capability of substrate node c_w
B_w	SCQ	Provided reliability of substrate node c_w
$\mu_{O_m^k}$, $\sigma_{O_m^k}$	GUR	Mean, STD of required resource of a_m^k in N_k
$\mu_{D_m^k}$, $\sigma_{D_m^k}$	SUR	Mean, STD of required computation capacity of a_m^k in N_k
$\mu_{M_m^k}$, $\sigma_{M_m^k}$	SUR	Mean, STD of required memory capacity of a_m^k in N_k
W_m^k	GCQ	Required capability of a_m^k in N_k
B_m^k	SCQ	Required reliability of a_m^k in N_k

The Tables 5.1 and 5.2 use the following abbreviations to categorize the parameters:

- G for general versus S for specific
- C for certain versus U for uncertain
- R for resource versus Q for capability (or quality)

This results in a three-letter acronym for the classification of each parameter, for instance, GCR for a general certain resource.

Table 5.2: Certain and Uncertain Link Resources and Capabilities

Def.	**Type**	**Description**
μ_{R_j}, σ_{R_j}	GUR	Mean, STD of provided resource of substrate edge e_j
μ_{T_j}, σ_{T_j}	SUR	Mean, STD of provided throughput of substrate edge e_j
Q_j	GCQ	Provided capability of substrate edge e_j
L_j	SCQ	Latency on substrate edge e_j
A_j	SCQ	Reliability of substrate edge e_j
$\mu_{R_i^k}$, $\sigma_{R_i^k}$	GUR	Mean, STD of required resource of link l_i^k in N_k
$\mu_{T_i^k}$, $\sigma_{T_i^k}$	SUR	Mean, STD of required throughput of link l_i^k in N_k
Q_i^k	GCQ	Required capability of link l_i^k in N_k
L_i^k	SCQ	Required latency of link l_i^k in N_k
A_i^k	SCQ	Required reliability of link l_i^k in N_k

The robust NSE is subject to numerous quality of service constraints, for instance, the throughput and reliability of the communication links and the computation power and memory of the cloud nodes.
The expected available throughput of an edge e_j in the substrate is represented by a normal distribution with a mean μ_{T_j} and an STD σ_{T_j}. For simplicity, the uplink and downlink data traffic is combined to one throughput parameter in this model. The probability distribution accounts for fluctuations in the signal quality, which results in a varying available throughput. For example, the SNIR and therefore the channel quality as well as the actual throughput in the RAN highly depend on, e.g., the distance and obstacles between the UE and the antenna as well as weather conditions and interferences.
The link latency L_j of e_j is assumed to be constant. However, in practice the link latency only remains constant as long as the link throughput capacity is not exceeded and a congestion in data traffic causes an additional delay.
Furthermore, the cloud servers $c_w \in \mathcal{C}$ in the substrate have a constant computation power D_w and memory capacity M_w. In addition, the edges e_j as well as the nodes c_w have specific reliabilities A_j and B_w.
The NSIs require a specific maximum latency L_i^k for each communication link l_i^k. The required throughput, however, is uncertain and therefore modeled as a normal distribution $\mathcal{N}(\mu_{T_i^k}, \sigma_{T_i^k})$ for each link $l_i^k \in \mathcal{L}_k$. Note that an STD of zero represents the special case of resource certainty.
The required computation power and the memory capacity for the applications are also defined as normal distributions $\mathcal{N}(\mu_{D_m^k}, \sigma_{D_m^k})$ and $\mathcal{N}(\mu_{M_m^k}, \sigma_{M_m^k})$.
Finally, the mapping of the NSIs must respect a predefined link reliability A_i^k as well

as a node reliability B^k_m.

The free-self-links, see Definition 6, as well as the free-self-paths, see Definition 7, provide unlimited resources and perfect capabilities without uncertainties. That means, the POF of a virtual communication link mapped on a free-self-path is always equal to 1.

5.3.2 Objective Function

The uncertainty-aware objective function is a key-component of the robust NSE model.

Definition 40 (Uncertainty-Aware Objective Function)
The robust NSE objective function is defined as

$$\max_{y_k, a2c^k_{m,w}, l2e^k_{i,j}} \left(f_{rev}\left(y_k\right) + \rho_1 \cdot f_{thr}\left(l2e^k_{i,j}\right) + \rho_2 \cdot f_{cpu}\left(a2c^k_{m,w}\right) + \rho_3 \cdot f_{mem}\left(a2c^k_{m,w}\right)\right)$$

with

$$f_{rev}\left(y_k\right) := \sum_{\forall k} y_k \frac{\omega_k}{\min\limits_k \omega_k}$$

$$f_{thr}\left(l2e^k_{i,j}\right) := -\sum_{\forall k,i,j} l2e^k_{i,j} \cdot \frac{\mu_{T^k_i} + \sigma_{T^k_i}}{\max\left(\mu_{T_j} - \sigma_{T_j}, \epsilon\right) \cdot \beta_{thr}}$$

$$f_{cpu}\left(a2c^k_{m,w}\right) := -\sum_{\forall k,m,w} a2c^k_{m,w} \cdot \frac{\mu_{D^k_m} + \sigma_{D^k_m}}{D_w \cdot \beta_{cpu}}$$

$$f_{mem}\left(a2c^k_{m,w}\right) := -\sum_{\forall k,m,w} a2c^k_{m,w} \cdot \frac{\mu_{M^k_m} + \sigma_{M^k_m}}{M_w \cdot \beta_{mem}}$$

and

$$\beta_{thr} := \sum_{\forall k,i,j} \frac{\mu_{T^k_i} + \sigma_{T^k_i}}{\max\left(\mu_{T_j} - \sigma_{T_j}, \epsilon\right)}$$

$$\beta_{cpu} := \sum_{\forall k,m,w} \frac{\mu_{D^k_m} + \sigma_{D^k_m}}{D_w}$$

$$\beta_{mem} := \sum_{\forall k,m,w} \frac{\mu_{M^k_m} + \sigma_{M^k_m}}{M_w}$$

$$\epsilon := 1 \cdot 10^{-10}$$

$$\rho_1 + \rho_2 + \rho_3 = 1$$

The uncertainty-aware objective function assumes that every embedded NSI-R contributes a positive revenue. Thus, the first priority of the uncertainty-aware objective function is to embed as many NSI-Rs as possible without violating the required capabilities, resource requirements and safety buffers. If there are not enough resources to accept all NSI-Rs, NSI-Rs with higher revenues are preferred over NSI-Rs with lower revenues. This is achieved by using the simple objective function from Definition 16 as the first term of the uncertainty-aware objective function. Since f_{rev} uses the normalization term $\frac{1}{\min\limits_{k} \omega_k}$, the revenue of the least beneficial NSI is scaled to 1 and the other weights respect the relative importances given by the original weights. Consequently, every embedded NSI makes a contribution of at least 1 to the objective value.
The f_{rev}-term selects the NSI-Rs to be embedded into the substrate network. The remaining three terms are responsible for optimizing the mapping of the elements of the embedded NSI-Rs on the elements of the substrate network with respect to minimizing uncertainty.
In the f_{thr}, f_{cpu} and f_{mem} terms the uncertainty of the availability of the respective resources on the allocated physical network elements is minimized. This is achieved by subtracting a relatively small penalty from f_{rev} depending on the uncertainty resulting from each mapping. This way, the optimization algorithm minimizes the overall uncertainty.
f_{thr} minimizes the sum of relative throughput utilizations, including the STDs. If the mean throughput of a physical link is smaller than or equal to its STD, then a small ϵ is used to prevent dividing by zero or by negative values. $\epsilon > 0$ is defined as a very small positive double value. $\epsilon = 1 \cdot 10^{-10}$ has been used in the evaluation of this thesis. The term f_{thr} is normalized by the factor β_{thr}. Since, the overall optimization problem is formalized as a maximization problem, f_{thr} is subtracted in the uncertainty-aware objective function.
Similarly, the terms f_{cpu} and f_{mem} are responsible for minimizing the shares of the allocated computation power and memory, plus the respective STDs.
The weights ρ_1, ρ_2 and $\rho_3 \in (0, 1)$ associated with the three penalty-terms sum up to one. Consequently, the normalized penalties for uncertainty cannot exceed the revenue of 1 of embedding the least beneficial NSI. Thus, accepting as many NSI-Rs as feasible dominates the requirement of minimizing uncertainty in the objective function.
Throughout this thesis, equal importance of the uncertain resources is assumed. Therefore, the weights are set to $\rho_1 = \rho_2 = \rho_3 = \frac{1}{3}$.

Referring to Definition 15, the $l2e$ variables in the uncertainty-aware objective function in Definition 40 are transformed into the corresponding $l2p$ variables.

$$\max_{y_k, a2c^k_{m,w}, l2p^k_{i,r}} \left(f_{rev}\left(y_k\right) + \rho_1 \cdot f_{thr}\left(l2p^k_{i,r}\right) + \rho_2 \cdot f_{cpu}\left(a2c^k_{m,w}\right) + \rho_3 \cdot f_{mem}\left(a2c^k_{m,w}\right) \right)$$

with

$$f_{thr}\left(l2p_{i,r}^{k}\right) := -\sum_{\forall k,i,j}\left(\sum_{\forall r} l2p_{i,r}^{k}\cdot p2e_{r,j}\right)\cdot\frac{\mu_{T_i^k}+\sigma_{T_i^k}}{\max\left(\mu_{T_j}-\sigma_{T_j},\epsilon\right)\cdot\beta_{thr}}$$

β_{thr}, f_{cpu} and f_{mem} remain unchanged, as in Definition 40.

5.3.3 Constraints

Based on one of the four NSE models, presented in Chapter 4, the robust NSE model is developed. Besides the uncertainty-aware objective functions, only the constraints regarding uncertain resources and capabilities have to be modified. All other constraints remain the same. This allows the application of the robust NSE on the basic, path-splitting, MAI and combined NSE model.

The general link resource constraint for an uncertain link resource R is defined as:

Definition 41 (General Uncertain Link Resource Constraints)
The uncertain link resource constraints in the robust NSE model are defined as

$$\sum_{\forall k,i} l2e_{i,j}^{k}\cdot(\mu_{R_i^k}+\gamma\cdot\sigma_{R_i^k})\leq(\mu_{R_j}-\gamma\cdot\sigma_{R_j}),\ \forall j$$

The general uncertain link resource constraints require that the sum of all expected demands of this resources increased by a safety buffer must not exceed the expected resource provisioning minus a safety discount. The safety discount/buffer is defined as a factor γ multiplied with the STD of the respective resource availability or demand probability distribution. A high γ can improve the robustness of the embedding, whereas a low γ might result in a higher benefit for the network operator, but requires a higher risk tolerance. If $\gamma = 1$, then the resources buffer equals the expected deviation in resource demand or provisioning respectively, that means, the STD. A higher γ provides a higher robustness. In the following evaluation γ is set to 1.5. A detailed analysis of the resulting POF with respect to the selection of the factor γ is provided in Section 5.3.5.

The general link resource constraints for uncertain link resources can be transformed to the context of an $l2p$ mapping by using Definition 15.

$$\sum_{\forall k,i}\left(\sum_{\forall r} l2p_{i,r}^{k}\cdot p2e_{r,j}\right)\cdot(\mu_{R_i^k}+\gamma\cdot\sigma_{R_i^k})\leq\left(\mu_{R_j}-\gamma\cdot\sigma_{R_j}\right),\ \forall j$$

The specific example for a link resource analyzed in this thesis is the throughput. The throughput constraints in the robust NSE are defined as follows.

Definition 42 (Uncertain Throughput Constraints)
The uncertain throughput constraints in the robust NSE model are defined as

$$\sum_{\forall k,i} l2e_{i,j}^{k}\cdot(\mu_{T_i^k}+\gamma\cdot\sigma_{T_i^k})\leq(\mu_{T_j}-\gamma\cdot\sigma_{T_j}),\ \forall j$$

The uncertain throughput constraints require, that the demanded throughput of the virtual links assigned to a physical edge plus the safety buffer must not exceed the provided resources minus a safety discount.

These constraints can be broken down to the associated link to path mappings, by doing the above explained transformation with Definition 15.

$$\sum_{\forall k,i} \left(\sum_{\forall r \text{ with } e_j \in P_r} l2p_{i,r}^k \right) \cdot (\mu_{T_i^k} + \sigma_{T_i^k}) \leq (\mu_{T_j} - \gamma \cdot \sigma_{T_j}),\ \forall j$$

Similarly, the node resource constraints are adapted for the robust NSE model. For an arbitrary uncertain node resource O we define the following set of constraints.

Definition 43 (General Uncertain Node Resource Constraints)
The uncertain node resource constraints in the robust NSE model are defined as

$$\sum_{\forall k,m} a2c_{m,w}^k \cdot (\mu_{O_m^k} + \gamma \cdot \sigma_{O_m^k}) \leq O_w,\ \forall w$$

For the node resources it is assumed that the provided resources are certain, i.e., they do not underlie uncertainty. Therefore, only a safety buffer is added to the resource demands. The sum of these demands increased by the safety buffer $\gamma \cdot \sigma_{O_m^k}$ must not exceed the provided resources of the physical node they are mapped on.

The CPU and memory resource constraints are derived from Definition 43.

Definition 44 (Uncertain CPU Constraints)
The uncertain CPU constraints in the robust NSE model are defined as

$$\sum_{\forall k,m} a2c_{m,w}^k \cdot (\mu_{D_m^k} + \gamma \cdot \sigma_{D_m^k}) \leq D_w,\ \forall w$$

Definition 45 (Uncertain Memory Constraints)
The uncertain memory constraints in the robust NSE model are defined as

$$\sum_{\forall k.m} a2c_{m,w}^k \cdot (\mu_{M_m^k} + \gamma \cdot \sigma_{M_m^k}) \leq M_w,\ \forall w$$

5.3.4 Robust NSE Model

This subsection provides an overview over the robust NSE model.
The new constraints of the robust NSE model are summarized. The remaining constraints number 4 to 12 are taken from one of the models from Chapter 4, either the basic, path-splitting, MAI or combined NSE model.

Model 5 (Robust NSE Model)

$$\max_{y_k, a2c^k_{m,w}, l2e^k_{i,j}} \left(\sum_{\forall k} y_k \frac{\omega_k}{\min\limits_k \omega_k} - \rho_1 \cdot \sum_{\forall k,i,j} l2e^k_{i,j} \cdot \frac{\mu_{T^k_i} + \sigma_{T^k_i}}{\max\left(\mu_{T_j} - \sigma_{T_j}, \epsilon\right) \cdot \beta_{thr}} \right.$$

$$\left. - \rho_2 \cdot \sum_{\forall k,m,w} a2c^k_{m,w} \cdot \frac{\mu_{D^k_m} + \sigma_{D^k_m}}{D_w \cdot \beta_{cpu}} - \rho_3 \cdot \sum_{\forall k,m,w} a2c^k_{m,w} \cdot \frac{\mu_{M^k_m} + \sigma_{M^k_m}}{M_w \cdot \beta_{mem}} \right)$$

under

1. *Uncertain Throughput Constraints:*

$$\sum_{\forall k,i} l2e^k_{i,j} \cdot (\mu_{T^k_i} + \gamma \cdot \sigma_{T^k_i}) \leq (\mu_{T_j} - \gamma \cdot \sigma_{T_j}),\ \forall j$$

2. *Uncertain CPU Constraints:*

$$\sum_{\forall k,m} a2c^k_{m,w} \cdot (\mu_{D^k_m} + \gamma \cdot \sigma_{D^k_m}) \leq D_w,\ \forall w$$

3. *Uncertain Memory Constraints:*

$$\sum_{\forall k,m} a2c^k_{m,w} \cdot (\mu_{M^k_m} + \gamma \cdot \sigma_{M^k_m}) \leq M_w,\ \forall w$$

The constraints number 4 to 12 are taken from the basic, path-splitting, MAI or combined path-splitting and MAI NSE model.

5.3.5 Network Slice Instance Acceptance Metrics

The model as described above is used to determine a nearly optimal NSE using the most reliable network resources.
Using the worst-case resource demands and availabilities is not beneficial, since it would require a significant resource overprovisioning of scarce resources, like RAN throughput. In order to provide a beneficial solution, the expected resource demands and provisioning are used as a basis for the robust NSE embedding. This can lead to a resource overbooking and resource availability violations can occur.
In this section, metrics for the resource availability and overbooking analysis are presented.

Probability of Resource Feasibility The probability of meeting the resource constraints is calculated to assess the probability of SLA compliance, or the risk of SLA violation respectively. This evaluation is done for each uncertain resource, while the node and link resources are distinguished.
As mentioned before, the provided node resources are fixed and do not underlie

uncertainty, while the node resource demands are assumed to be uncertain and underlie a normal distribution. That means, it is expected that the actually provided amount of resources is probably close to the expected value.
The following metrics can also be applied on uncertain resources with different distributions, other than the normal distribution. Any closed or discrete probabilistic distribution function can be used. Non the less, some mathematical conversions in the following calculation are tailored to Gaussian distributed uncertain resources.

A fixed node resource O_w, provided by a specific substrate node c_w, as well as uncertain resource demands of virtual application nodes a_m^k with Gaussian distributions $\mathcal{N}(\mu_{O_m^k}, \sigma_{O_m^k})$ are given. The overall demand of the general node resource O is the sum of the normal distributed density functions. By convolution of the normal distributions the result is again a normal distribution.

$$\mathcal{N}\left(\sum_{\forall k,m} a2c_{m,w}^k \cdot \mu_{O_m^k}, \sqrt{\sum_{\forall k,m} a2c_{m,w}^k \cdot (\sigma_{O_m^k})^2}\right)$$

The stochastic resource demand of an application a_m^k is only considered in the sum of the demands on a specific physical node c_w, if a_m^k is mapped on c_w, i.e., if $a2c_{m,w}^k = 1$.

The residual resources of c_w can be calculated by convolution of the normal distributions.

$$\mathcal{N}\left(O_w - \sum_{\forall k,m} a2c_{m,w}^k \cdot \mu_{O_m^k}, \sqrt{\sum_{\forall k,m} a2c_{m,w}^k \cdot (\sigma_{O_m^k})^2}\right)$$

This is the probability density function of the residual resources on the cloud node c_w in the substrate network, considering the allocated virtual application nodes with their uncertain resource demands.

Regarding the assessment of the probability of SLA compliance, we define the so-called POF of general cloud node resources.

Definition 46 (POF of General Node Resources)
The POF of a general, uncertain cloud node resources O for a cloud node c_w is defined as

$$POF_{O_w} := \int_0^\infty \mathcal{N}\left(O_w - \sum_{\forall k,m} a2c_{m,w}^k \cdot \mu_{O_m^k}, \sqrt{\sum_{\forall k,m} a2c_{m,w}^k \cdot (\sigma_{O_m^k})^2}\right) dx$$

I.e., the POF for meeting the constraint requirements for the general node resource O for the embedded NSI-Rs is represented by the probability that the residual resources are greater or equal to zero.

Based on this general node resource assessment, the CPU and memory resources are assessed on a per physical node basis. The respective POFs are defined as follows.

Definition 47 (POF of CPU Node Resources)
The POF of the uncertain CPU resources D for a cloud node c_w is defined as

$$POF_{D_w} := \int_0^\infty \mathcal{N}\left(D_w - \sum_{\forall k,m} a2c^k_{m,w} \cdot \mu_{D^k_m}, \sqrt{\sum_{\forall k,m} a2c^k_{m,w} \cdot (\sigma_{D^k_m})^2}\right) dx$$

Definition 48 (POF of Memory Node Resources)
The POF of the uncertain memory resources M for a cloud node c_w is defined as

$$POF_{M_w} := \int_0^\infty \mathcal{N}\left(M_w - \sum_{\forall k,m} a2c^k_{m,w} \cdot \mu_{M^k_m}, \sqrt{\sum_{\forall k,m} a2c^k_{m,w} \cdot (\sigma_{M^k_m})^2}\right) dx$$

The POF_{D_w} and POF_{M_w} are determined for every cloud node c_w.

For the link resources we assume that both, the provided resources as well as the required resources, can be subject to uncertainty. For modeling the provided general link resources on the physical links we use a normal distributed with $\mathcal{N}\left(\mu_{R_j}, \sigma_{R_j}\right)$). The general link resource required by a virtual link R^k_i are assumed to be normal distributed with $\mathcal{N}\left(\mu_{R^k_i}, \sigma_{R^k_i}\right)$. By convolution of the normal distributions we receive the sum of demands on a per physical link basis.

$$\mathcal{N}\left(\sum_{\forall k,i} l2e^k_{i,j} \cdot \mu_{R^k_i}, \sqrt{\sum_{\forall k,i} l2e^k_{i,j} \cdot (\sigma_{R^k_i})^2}\right)$$

When a virtual link l^k_i is mapped on a physical edge e_j, i.e., when $l2e^k_{i,j} > 0$, then it is considered in the sum of the demands. If path-splitting is enabled the NSE algorithm determines an $l2e^k_{i,j} \in [0,1]$. Then the expected resource utilization for l^k_i is weighted with the proportion of the usage of $l2e^k_{i,j}$ before being considered in the sum of resource demands.

The probability density function of the residual general link resources is

$$\mathcal{N}\left(\mu_{R_j} - \sum_{\forall k,i} l2e^k_{i,j} \cdot \mu_{R^k_i}, \sqrt{(\sigma_{R_j})^2 + \sum_{\forall k,i} l2e^k_{i,j} \cdot (\sigma_{R^k_i})^2}\right)$$

The POF of the general link resource is defined as follows.

Definition 49 (POF of General Link Resources)
The POF of a general, uncertain communication link resources R for a physical edge e_j is defined as

$$POF_{R_j} := \int_0^\infty \mathcal{N}\left(\mu_{R_j} - \sum_{\forall k,i} l2e^k_{i,j} \cdot \mu_{R^k_i}, \sqrt{(\sigma_{R_j})^2 + \sum_{\forall k,i} l2e^k_{i,j} \cdot (\sigma_{R^k_i})^2}\right) dx$$

This is applied to the throughput resources.

Definition 50 (POF of Throughput Resources)
The POF of the uncertain throughput resources T for a physical edge e_j is defined as

$$POF_{T_j} := \int_0^\infty \mathcal{N}\left(\mu_{T_j} - \sum_{\forall k,i} l2e_{i,j}^k \cdot \mu_{T_i^k}, \sqrt{(\sigma_{T_j})^2 + \sum_{\forall k,i} l2e_{i,j}^k \cdot (\sigma_{T_i^k})^2}\right) dx$$

When analyzing the overall confidence of a resource type within an NSI-R as well as the overall POF of an NSI, stochastic independence is assumed. Then the POFs of the elements can simply be multiplied to determine the overall POF of each resource and a whole NSI. I.e., it is assumed that an overload on an element of the physical network is independent of an overload on other elements. In reality, these incidents are often dependent on each other. However, assuming stochastic independence is assuming the worst-case scenario, since dependent overload on several elements would increase the overall POF.

The combined POFs of the resources are specified in the Definitions 51, 52 and 53.

Definition 51 (Combined POF of Throughput Resources)
The overall POF of the uncertain throughput resources T across all links in an NSI-R is defined as

$$POF_T^k := \prod_{\forall i,j \ with \ l2e_{i,j}^k > 0} POF_{T_j}, \forall k$$

Definition 52 (Combined POF of CPU Resources)
The overall POF of the uncertain CPU resources D across all nodes in an NSI-R is defined as

$$POF_D^k := \prod_{\forall m,w \ with \ a2c_{m,w}^k = 1} POF_{D_w}, \forall k$$

Definition 53 (Combined POF of Memory Resources)
The overall POF of the uncertain memory resources M across all nodes in an NSI-R is defined as

$$POF_M^k := \prod_{\forall m,w \ with \ a2c_{m,w}^k = 1} POF_{M_w}, \forall k$$

Finally, the overall confidence in the resource availability is determined. Under the assumption of stochastic independence of the different resource types, the POF is accumulated per NSI-R by multiplication of the resource POF of the NSI.

Definition 54 (Combined POF of NSI-R)
The overall POF of the uncertain resources of an NSI-R is defined as

$$POF^k := POF_T^k \cdot POF_D^k \cdot POF_M^k, \forall k$$

Since Definition 54 is based on the assumption of stochastic independence of overload of different resources POF^k is the worst-case POF.

Safety Buffer The safety buffer γ directly impacts the NSI-R feasibility. Using a higher γ improves the POFs of the accepted NSI-Rs on the one hand, but on the other hand sets higher requirements on the feasibility of an NSI-R. The numerical dependency between the γ and the POF of the accepted NSIs is set out below.

When considering a single resource, the POF is dependent on γ. For instance, for the general link resource constraint for a physical link e_j, the following equation from Definition 41 applies.

$$\sum_{\forall k,i} l2e^k_{ij} \cdot (\mu_{R^k_i} + \gamma \cdot \sigma_{R^k_i}) \le (\mu_{R_j} - \gamma \cdot \sigma_{R_j})$$

$$\Leftrightarrow$$

$$(\mu_{R_j} - \gamma \cdot \sigma_{R_j}) - \left(\sum_{\forall k,i} l2e^k_{i,j} \cdot (\mu_{R^k_i} + \gamma \cdot \sigma_{R^k_i}) \right) \ge 0$$

$$\Leftrightarrow$$

$$\left(\mu_{R_j} - \sum_{\forall k,i} l2e^k_{i,j} \cdot \mu_{R^k_i} \right) - \gamma \cdot \left(\sigma_{R_j} + \sum_{\forall k,i} l2e^k_{i,j} \cdot \sigma_{R^k_i} \right) \ge 0$$

$$\Leftrightarrow$$

$$\frac{\mu_{R_j} - \sum_{\forall k,i} l2e^k_{i,j} \cdot \mu_{R^k_i}}{\sigma_{R_j} + \sum_{\forall k,i} l2e^k_{i,j} \cdot \sigma_{R^k_i}} \ge \gamma$$

The nominator of this inequation is the expected amount of residual resources, while the denominator represents the cumulated expected deviations from the mean of the provided as well as demanded resources. The safety buffer γ provides a lower bound for the ratio of the expected free resources and the expected deviation. For instance, if $\gamma = 1.5$, then 50% more resources than the expected deviation in demand and provisioning are reserved as a resource buffer. A γ of 1 means that the residual resources must at least be equal to the accumulated expected deviations.
This can be transferred one-to-one to throughput and other similar link resources.

For the general node resources, the following equation from Definition 43 applies. The following inequation can be transferred one-to-one to the CPU, memory as wells other similar node resources for a cloud node c_w.

$$\sum_{\forall k,m} a2c^k_{m,w} \cdot (\mu_{O^k_m} + \gamma \cdot \sigma_{O^k_m}) \le O_w$$

$$\Leftrightarrow$$

$$\frac{O_w - \sum_{\forall k,m} a2c^k_{m,w} \cdot \mu_{O^k_m}}{\sum_{\forall k,m} a2c^k_{m,w} \cdot \sigma_{O^k_m}} \ge \gamma$$

Thus, the worst-case POF of a link or node resource can be directly controlled by the selection of γ.

For better readability, we define the following notations.

Definition 55 (Residual Mean and STD of Link Resources)
For the general link resource R, we define the residual mean and STD as

$$\mu_{resL} := \mu_{R_j} - \sum_{\forall k,i} l2e^k_{i,j} \cdot \mu_{R^k_i}$$

$$\sigma_{resL} := \sigma_{R_j} + \sum_{\forall k,i} l2e^k_{i,j} \cdot \sigma_{R^k_i}$$

Definition 56 (Residual Mean and STD of Node Resources)
For the general node resource O, we define the residual mean and STD as

$$\mu_{resN} := O_w - \sum_{\forall k,m} a2c^k_{m,w} \cdot \mu_{O^k_m}$$

$$\sigma_{resN} := \sum_{\forall k,m} a2c^k_{m,w} \cdot \sigma_{O^k_m}$$

With Definition 55 the link resource constraints (see Definition 41) can be simplified to

$$\mu_{resL} \geq \gamma \cdot \sigma_{resL}$$

and with Definition 56 the node resource constraints (see Definition 43) can be written as

$$\mu_{resN} \geq \gamma \cdot \sigma_{resN}$$

That means, the residual link and node resources must be equal to or higher than the assoicated accumulated required resource buffers.

With these definitions, the worst-case POFs of the general link resources R_j can be written as

$$POF_{R_j} \leq 1 - \Phi(\gamma) = 1 - \left(\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\gamma} e^{-\frac{1}{2}x^2} dx \right)$$

The same applies to the node resource POFs.

$$POF_{O_w} \leq 1 - \Phi(\gamma)$$

Figure 5.3 provides a graphical illustration of the worst-case POF and its interdependency with γ. The gray space under the blue density function of the normal distribution represents the POF of a mapping with the highest allowed uncertainty. The mean of the convoluted normal distributions is equal to $\gamma \cdot \sigma_{resL}$ or $\gamma \cdot \sigma_{resN}$ respectively. The space under the normal distribution on the right-hand side of the mean equals 0.5 or 50%.

The worst-case POF is independent of the residual mean and STDs, it only depends on γ. The worst-case POF of single elements and common values for γ can be derived from the Table 5.3. They are to be interpreted as the worst possible POF of any of the link or node mappings. The aggregated POF of a resource type or for an NSI can drop below these values, due to multiplication of the individual POF values. However, in most practical cases, the actual POFs exceed the worst-case values.

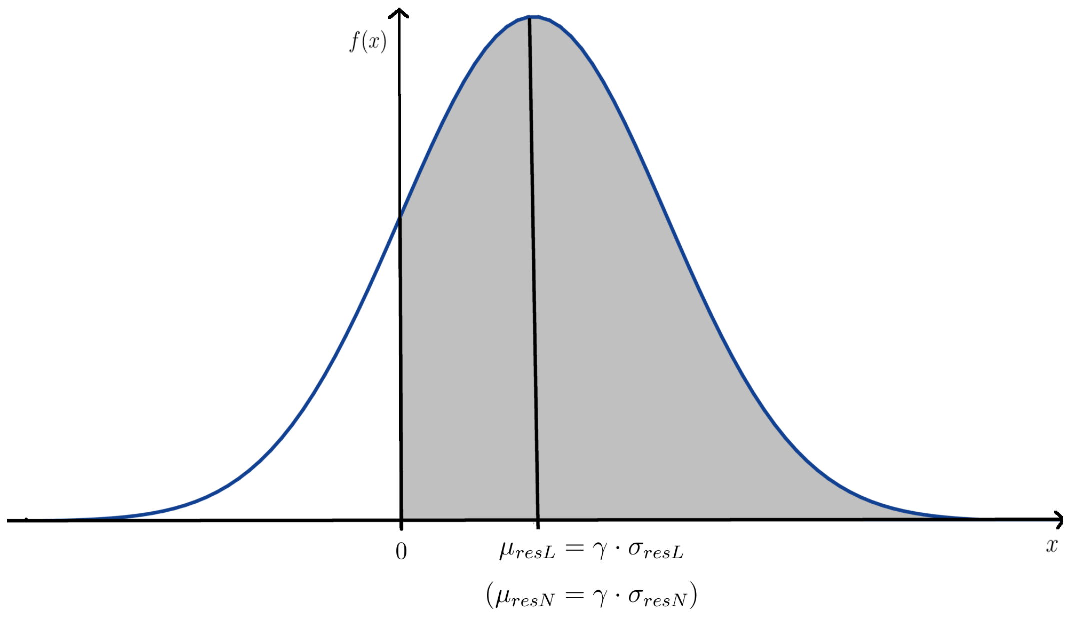


Figure 5.3: Worst-Case POF

Table 5.3: Safety Discount and Worst-Case POF

γ	Worst-Case POF	γ	Worst-Case POF	γ	Worst-Case POF
0.0	50.00000	1.0	84.13447	2.0	97.72499
0.1	53.98278	1.1	86.43339	2.1	98.21356
0.2	57.92597	1.2	88.49303	2.2	98.60966
0.3	61.79114	1.3	90.31995	2.3	98.92759
0.4	65.54217	1.4	91.92433	2.4	99.18025
0.5	69.14625	1.5	93.31928	2.5	99.37903
0.6	72.57469	1.6	94.52007	2.6	99.53388
0.7	75.80363	1.7	95.54345	2.7	99.65330
0.8	78.81446	1.8	96.40697	2.8	99.74449
0.9	81.59399	1.9	97.12834	2.9	99.81342
3.0	99.86501	4.0	99.99683	5.0	99.99997

Example In the following, NSE under uncertainty is illustrated by means of a simple example.
Given a small substrate network with only one UE group and two connected cloud nodes, two minimal NSI-Rs should be embedded considering uncertainties in resource provisioning and demands. The NSI weights are equal for the two NSI-Rs: $\omega_0 = 1$ and $\omega_1 = 1$.
Figure 5.4 provides a graphical representation of the substrate network and Figure 5.5 as well as Figure 5.6 illustrate the NSI-Rs.

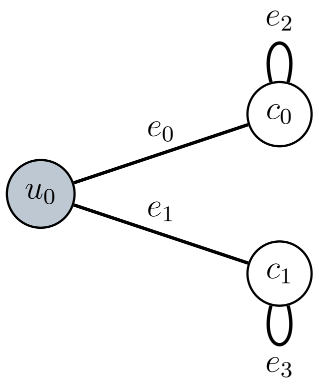


Figure 5.4: Uncertain Example - Substrate Network

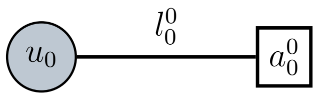


Figure 5.5: Uncertain Example - Network Slice 0

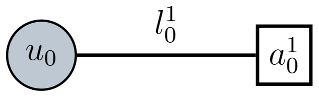


Figure 5.6: Uncertain Example - Network Slice 1

The provided and required resources are summarized in the Tables 5.4, 5.5 and 5.6.

Table 5.4: Uncertain Example - Network Node Parameters Substrate

Node	D	M	A
c_0	11.5	20	0.95
c_1	15	15	0.95

This NSE problem is solved in its basic form (no path-splitting and no MAI). The penalty weights for the throughput, CPU and memory resources are set to $\rho_1 = \rho_2 = \rho_3 = \frac{1}{3}$.

Table 5.5: Uncertain Example - Network Node Parameters NSIs

Node	μ_D	σ_D	μ_M	σ_M	A
a_0^0	10	1	10	1	0.95
a_0^1	10	1	10	1	0.95

Table 5.6: Uncertain Example - Network Link Parameters

Link	μ_T	σ_T	L	A
e_0	9	2	1	0.95
e_1	10	2	1	0.95
e_2	∞	0	0	1.00
e_3	∞	0	0	1.00
l_0^0	2	2	1	0.95
l_0^1	3	2	1	0.95

The binary NSI mapping variables y_0 and y_1 are defined. Additionally, the binary $a2c$ variables are listed in Table 5.7 and the binary $l2p$ variables are listed in Table 5.8.

Table 5.7: Uncertain Example - Application to Cloud Node Mapping Variables

	a_0^0	a_0^1
c_0	$a2c_{0,0}^0$	$a2c_{0,0}^1$
c_1	$a2c_{0,1}^0$	$a2c_{0,1}^1$

The $l2p$ mappings are based on the following path definitions:

$$
\begin{aligned}
&P_0 := \{e_1\} \\
&P_1 := \{e_0\} \\
&P_2 := P_{c_0}^{free} = \{e_2\} \\
&P_3 := P_{c_1}^{free} = \{e_3\}
\end{aligned}
$$

Table 5.8: Uncertain Example - Link to Path Mapping Variables

	l_0^0	l_0^1
P_0	$l2p_{0,0}^0$	$l2p_{0,0}^1$
P_1	$l2p_{0,1}^0$	$l2p_{0,1}^1$
P_2	-	-
P_3	-	-

Note that the free-self paths are not feasible for any mapping since there is no application to application link in the NSIs.
The safety-buffer γ is set to 1.5 for this example.

Then, the uncertain resource constraints can be written as follows.

1. Throughput Constraints:
The throughput demands as well as the throughput provisioning are subject to uncertainty. Safety buffers are used, to improve the resource confidence of the embedding solution.

- Throughput capacity constraint for e_0:

  $(3 + 1.5 \cdot 2) \cdot l2p^1_{0,1} + (2 + 1.5 \cdot 2) \cdot l2p^0_{0,1} \leq 9 - 1.5 \cdot 2$
  $\Leftrightarrow 6 \cdot l2p^1_{01} + 5 \cdot l2p^0_{01} \leq 6$

- Throughput capacity constraint for e_1:

  $(3 + 1.5 \cdot 2) \cdot l2p^1_{00} + (2 + 1.5 \cdot 2) \cdot l2p^0_{00} \leq 10 - 1.5 \cdot 2$
  $\Leftrightarrow 6 \cdot l2p^1_{0,0} + 5 \cdot l2p^0_{0,0} \leq 7$

The throughput constraints allow exactly one virtual link to be mapped on every physical path in this example. Which virtual link is mapped on which physical path is subject to optimization.

The CPU as well as the memory capacity constraints are defined for each physical cloud node. The node resource demands are subject to uncertainty, while the resource provisioning is assumed to be certain.

2. CPU Constraints:
- CPU capacity constraint for c_0:

  $(10 + 1.5 \cdot 1) \cdot a2c^1_{0,0} + (10 + 1.5 \cdot 1) \cdot a2c^0_{0,0} \leq 11.5$
  $\Leftrightarrow 11.5 \cdot a2c^1_{0,0} + 11.5 \cdot a2c^0_{0,0} \leq 11.5$

- CPU capacity constraint for c_1:

  $(10 + 1.5 \cdot 1) \cdot a2c^1_{0,1} + (10 + 1.5 \cdot 1) \cdot a2c^0_{0,1} \leq 15$
  $\Leftrightarrow 11.5 \cdot a2c^1_{0,1} + 11.5 \cdot a2c^0_{0,1} \leq 15$

3. Memory Constraints:
- Memory capacity constraint for c_0:

  $(10 + 1.5 \cdot 1) \cdot a2c^1_{0,0} + (10 + 1.5 \cdot 1) \cdot a2c^0_{0,0} \leq 20$
  $\Leftrightarrow 11.5 \cdot a2c^1_{0,0} + 11.5 \cdot a2c^0_{0,0} \leq 20$

- Memory capacity constraint for c_1:

  $(10 + 1.5 \cdot 1) \cdot a2c^1_{0,1} + (10 + 1.5 \cdot 1) \cdot a2c^0_{0,1} \leq 15$
  $\Leftrightarrow 11.5 \cdot a2c^1_{0,1} + 11.5 \cdot a2c^0_{0,1} \leq 15$

The node mappings are restricted by the CPU and Memory resource such that only one virtual node (application) can be mapped on a physical node (cloud node). Overall there are enough resource to embed both NSIs. However, only the two option of mapping a^0_0 on c_0 and a^1_0 on c_1 or the other way around are feasible solutions for embedding both NSIs.

The optimal allocation minimizes the uncertainty. The uncertainty objective function provides insight into the optimization problem. The revenue term is as simple as:

$$f_{rev}(y_k) = 1 \cdot y_0 + 1 \cdot y_1$$

while the uncertainty penalties are defined as follows:

$$f_{thr}(l2e^k_{i,j}) = -\frac{2+2}{\max(9-2,\epsilon) \cdot \beta_{thr}} \cdot l2e^0_{0,0} - \frac{3+2}{\max(9-2,\epsilon) \cdot \beta_{thr}} \cdot l2e^1_{0,0}$$
$$-\frac{2+2}{\max(10-2,\epsilon) \cdot \beta_{thr}} \cdot l2e^0_{0,1} - \frac{3+2}{\max(10-2,\epsilon) \cdot \beta_{thr}} \cdot l2e^1_{0,1}$$
$$= -\frac{4}{7 \cdot \beta_{thr}} \cdot l2e^0_{0,0} - \frac{5}{7 \cdot \beta_{thr}} \cdot l2e^1_{0,0} - \frac{4}{8 \cdot \beta_{thr}} \cdot l2e^0_{0,1} - \frac{5}{8 \cdot \beta_{thr}} \cdot l2e^1_{0,1}$$

with

$$\beta_{thr} = \frac{4}{7} + \frac{5}{7} + \frac{4}{8} + \frac{5}{8} = \frac{135}{56}$$

and

$$f_{cpu}(a2c^k_{m,w}) = -\frac{10+1}{11.5 \cdot \beta_{cpu}} \cdot a2c^0_{0,0} - \frac{10+1}{15 \cdot \beta_{cpu}} \cdot a2c^0_{0,1} - \frac{10+1}{11.5 \cdot \beta_{cpu}} \cdot a2c^1_{0,0} - \frac{10+1}{15 \cdot \beta_{cpu}} \cdot a2c^1_{0,1}$$

$$= -\frac{11}{11.5 \cdot \beta_{cpu}} \cdot a2c^0_{0,0} - \frac{11}{15 \cdot \beta_{cpu}} \cdot a2c^0_{0,1} - \frac{11}{11.5 \cdot \beta_{cpu}} \cdot a2c^1_{0,0} - \frac{11}{15 \cdot \beta_{cpu}} \cdot a2c^1_{0,1}$$

with

$$\beta_{cpu} = \frac{11}{11.5} + \frac{11}{15} + \frac{11}{11.5} + \frac{11}{15} = \frac{1166}{345}$$

and

$$f_{mem}(a2c^k_{m,w}) = \frac{10+1}{20 \cdot \beta_{mem}} \cdot a2c^0_{0,0} - \frac{10+1}{15 \cdot \beta_{mem}} \cdot a2c^0_{0,1} - \frac{10+1}{20 \cdot \beta_{mem}} \cdot a2c^1_{0,0} - \frac{10+1}{15 \cdot \beta_{mem}} \cdot a2c^1_{0,1}$$

$$= -\frac{11}{20 \cdot \beta_{mem}} \cdot a2c^0_{0,0} - \frac{11}{15 \cdot \beta_{mem}} \cdot a2c^0_{0,1} - \frac{11}{20 \cdot \beta_{mem}} \cdot a2c^1_{0,0} - \frac{11}{15 \cdot \beta_{mem}} \cdot a2c^1_{0,1}$$

with

$$\beta_{mem} = \frac{11}{20} + \frac{11}{15} + \frac{11}{20} + \frac{11}{15} = \frac{77}{30}$$

The complete uncertainty-aware objective function, including the penalty weights ρ_1, ρ_2 and ρ_3, is transformed from the l2e to the l2p mapping.

$$\max_{y_k, a2c^k_{m,w}, l2p^k_{i,r}} (1 \cdot y_0 + 1 \cdot y_1 - 0.16577 \cdot a2c^0_{0,0} - 0.16577 \cdot a2c^1_{0,0} - 0.16757 \cdot a2c^0_{0,1} - 0.16757 \cdot a2c^1_{0,1}$$

$$-0.06914 \cdot l2p^0_{0,0} - 0.08642 \cdot l2p^1_{0,0} - 0.07901 \cdot l2p^0_{0,1} - 0.09877 \cdot l2p^1_{0,1})$$

Considering the resource constraints, the first option of mapping both NSIs is to map a^0_0 on c_0 and a^1_0 and c_1. This causes a penalty of 0.16577 for $a2c^0_{0,0}$ and 0.16757 for $a^1_{0,1}$. The corresponding link mapping uncertainty penalties are 0.07901 for $l2p^0_{0,1}$ and 0.08642 for $l2p^1_{0,0}$.
The second alternative allocation is mapping a^0_0 on c_1 and a^1_0 and c_0. The penalties for the uncertainties are 0.16757 for $a2c^0_{0,1}$, 0.16577 for $a2c^1_{0,0}$, 0.06914 for $l2p^0_{0,0}$ and 0.09877 for $l2p^1_{0,1}$.
Since the node resource requirements of the two application are the same, the accumulated node uncertainty penalties are equal. However, the accumulated link uncertainty penalties are smaller for the first alternative. Mapping a^0_0 on c_0 and a^1_0 and c_1 has an overall uncertainty penalty of 0.49877, while mapping a^0_0 on c_1 and a^1_0 and c_0 has a higher overall uncertainty penalty of 0.50125.
Thus, the best solution is the first mapping alternative. It is represented by the following non-zero variable assignments.

$$y_0 = 1 \qquad a2c^0_{0,0} = 1 \qquad l2p^0_{0,1} = 1$$
$$y_1 = 1 \qquad a2c^1_{0,1} = 1 \qquad l2p^1_{0,0} = 1$$

A graphical representation of this allocation is provided in Figure 5.7.

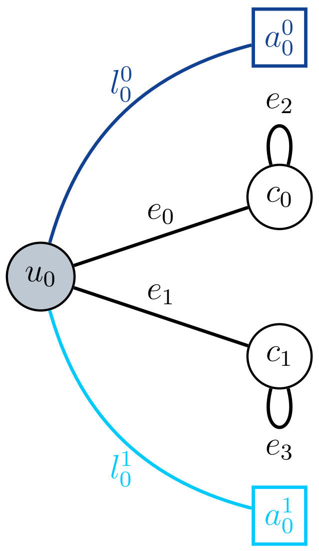


Figure 5.7: Uncertain Example - Uncertainty-Aware Optimization

For the confidence analysis of this simple example the POFs are calculated. The POFs of the substrate network elements and every resource, given the above NSI allocations, are:

- POFs of c_0:

$$POF_{D_0} = \int_0^\infty \mathcal{N}\,(11.5 - 10, 1) \approx 0.93319$$

$$POF_{M_0} = \int_0^\infty \mathcal{N}\,(20 - 10, 1) \approx 1.0$$

- POFs of c_1:

$$POF_{D_1} = \int_0^\infty \mathcal{N}\,(15 - 10, 1) \approx 0.9999997$$

$$POF_{M_1} = \int_0^\infty \mathcal{N}\,(15 - 10, 1) \approx 0.9999997$$

- POFs of e_0:

$$POF_{T_0} = \int_0^\infty \mathcal{N}\left(9 - 2, \sqrt{2^2 + 2^2}\right) \approx 0.99334$$

- POFs of e_1:

$$POF_{T_1} = \int_0^\infty \mathcal{N}\left(10 - 3, \sqrt{2^2 + 2^2}\right) \approx 0.99334$$

Every virtual element mapped on these physical network elements has the associated POF. There is only one virtual link and one application involved per NSI and the throughput, CPU and memory POFs per NSI are easy to determine.

$$POF_D^0 \approx 0.93319$$

$$POF_M^0 \approx 1.0$$

$$POF_T^0 \approx 0.99334$$

$$POF_D^1 \approx 0.9999997$$

$$POF_M^1 \approx 0.9999997$$

$$POF_T^1 \approx 0.99334$$

The combined POF per NSI-R is then:

$$POF^0 \approx 0.93319 \cdot 1.0 \cdot 0.99334 \approx 0.92697$$

$$POF^1 \approx 0.9999997 \cdot 0.9999997 \cdot 0.99334 \approx 0.99333$$

That means, the combined confidence, that the required resources of NSI-R 0 are available on request is approximately 92.697%. I.e., in 7.303% of the cases the resources are overbooked. The second NSI-R has a better POF of 99.333% and an expected overbooking rate of only approximately 0.667%.

6 Implementation and Evaluation

This chapter presents a comprehensive evaluation of the NSE solution and its variants, proposed in Chapters 4 and 5.
First of all, the evaluation setup is introduced. Secondly, the implementation and configuration used for this evaluation is described. In the third section, the effectiveness of path-splitting is evaluated. The subsequent section works out the dependency between the allowed latency of the NSIs and the use of MAI. Then, the combined path-splitting and MAI model is analyzed towards its runtime and effectiveness. Subsequently, the robust NSE is evaluated with regard to different safety buffers. Then, the runtime of the different variants of the NSE problem are compared and analyzed. Furthermore, their scalability is analyzed on NSE problems of different sizes. The final section of this chapter provides a summary and discussion of this evaluation.

6.1 Evaluation Setup

In this introductory section, the hardware used for this evaluation is explained. In addition, an overview over the prototypical implementation of the NSI Admission and Confidence Analysis Program, developed for this thesis, is provided. Finally, the evaluation scenarios are described in detail.

6.1.1 Hardware and Implementation

A prototypical implementation of the LP-based, nearly optimal NSE algorithms, introduced in this thesis, is used for this evaluation. The NSE algorithms as well as the scenario generation are implemented in form of a dedicated Java program. The software implements the models and algorithms proposed in the Chapters 4 and 5. The prototype of the so-called NSI Admission and Confidence Analysis Program consists of over 20 thousand lines of code, structured as shown in Figure 6.1. For solving the ILP the SCIP Optimization Suite 6.0 [49] is used.
The evaluation is executed on a normal notebook, a Mac Book Pro 2015 with a 3,1 GHz Intel Core i7 and a 16 GB 1867 MHz DDR3.

The NSI Admission and Confidence Analysis Program consists of three internal components: the NSE Test-Case Generator, the NSE-2-LP Converter and the Post-Processing.

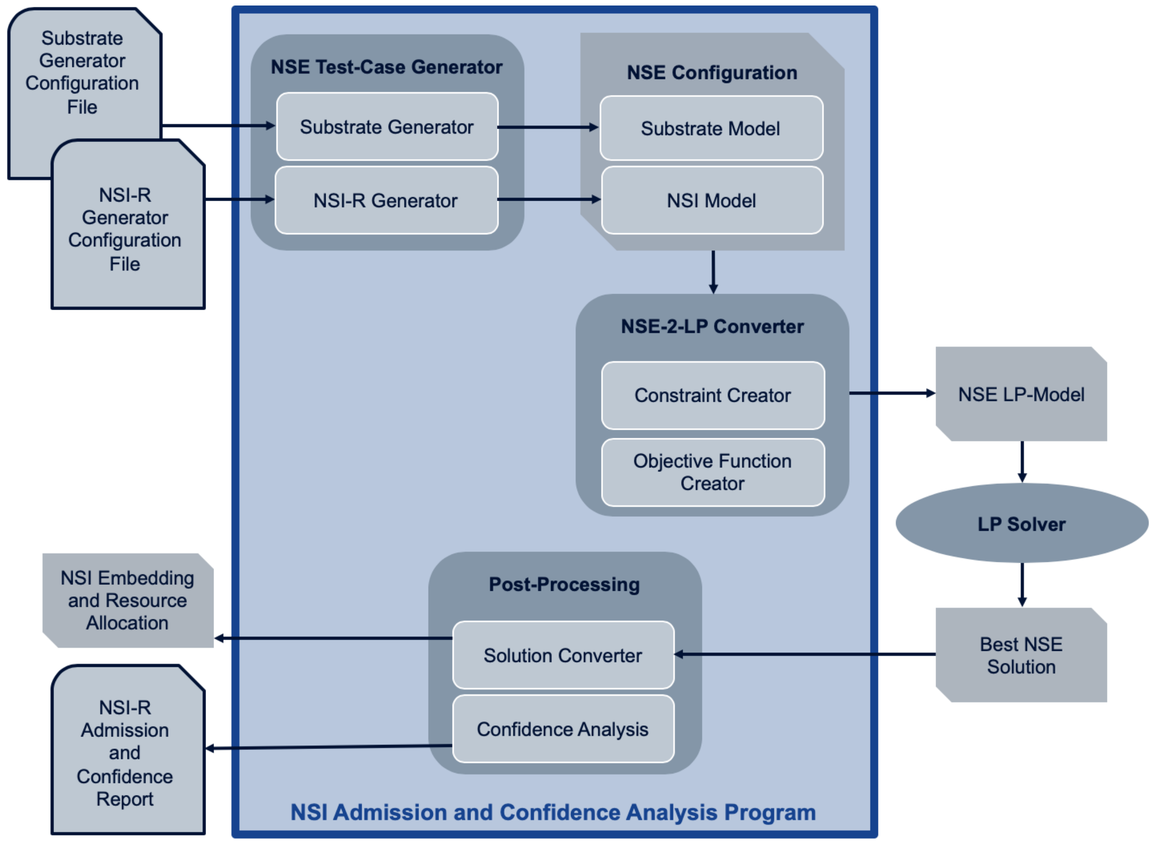


Figure 6.1: NSI Admission and Confidence Analysis Program

The NSE Test-Case Generator creates an NSE Configuration consisting of the Substrate Model and the NSI Models. For this purpose, a Substrate and an NSI-R Generator Configuration File is provided. These files are human-readable lists of configuration parameters with their values. More details on these configuration files are given in the following Section 6.1.2.
The NSE configuration is then transformed into the so-called NSE LP Model by the NSE-2-LP Converter. Hereby, the linear constraints as well as the objective function of the NSE problem are determined. The NSE problem is transformed into a generic ILP or MILP, when path-splitting is activated. This generic problem is then solved by an out-of-the-box solver. In this evaluation, the open source SCIP solver by Gleixner et al. [49] is used. Due to the modular structure of the program, the solver can easily be exchanged with a different one. For instance, an interface for the GLPK solver is part of the NSI Admission and Confidence Analysis Program. However, the GLPK is not efficient in finding solutions that are close to optimality for medium sized and big NSE problem, when path-splitting is allowed. The SCIP solver on the other hand is specialized on solving integer programs and mixed integer programs runtime efficiently. It is capable of finding the best solution in the basic, path-splitting as well as MAI and combined NSE problems.
The best solution calculated by the LP is interpreted by the Post-Processing module. Additionally, the confidence-analysis is executed in this module. The NSI Embedding and Resource Allocation as well as the NSI-R Admission and Confidence Report are the outputs of the NSI Admission and Confidence Analysis Program.

The Java interface of the SCIP solver provides two options for reading the constraints of the NSE LP Model. Either, all constraints can be provided in form of a vector of variables and a matrix of their parameter values, or in form of a vector of variables and the vectors of the single constraints. However, since the number of variables and constraints grows exponentially with the number of elements in the substrate network and in the NSIs, the required double-value parameter matrix exceeds the available main memory capacity by an order of magnitude, even for medium sized instances of the NSE problem. Although, most of the matrix entries are zero, since most constraints only concern a small share of the variables, all entries of the matrix have to be filled. Even using a more memory-efficient data-type for storing the matrix parameters cannot cope with the exponential growth of the matrix size for bigger instances of the NSE problem.
Thus, the linear constraints have to be inserted into the SCIP solver constraint by constraint, due to the memory restrictions of the hardware and the Java API of the SCIP solver. The same issue accounts for the GLPK solver as well. Thereby, transferring the constraints of the NSE LP model in the Java program to one of the solvers is very time consuming. The issue could be fixed with more efficient interfaces of the solvers.
Because of this, the runtime evaluation distinguishes between the preparation time and the time the solver takes to find the best solution, once prepared with the input data. This is referred to as the so-called embedding time.

6.1.2 Evaluation Scenarios

The evaluation scenarios used for this evaluation are randomly generated NSE problems, consisting of a substrate network with a star-topology with configurable size, resources and capabilities and a set of NSI-Rs with configurable parameters.

For the evaluation scenario generation two configuration files are created. The first configuration file describes the topology and characteristics of the substrate network. The second configuration file describes the number as well as the characteristics of the NSIs.

6.1.2.1 Physical Network

The substrate is generated randomly with the so-called Substrate Generator Configuration File. This file defines all necessary parameters for creating a new random physical network with a configurable size and topology as well as configurable characteristics, resources and capabilities. The file comprises the parameters, summarized in Table 6.1. Table 6.2 provides the associated default parameter values.

Table 6.1: Substrate Generator Configuration File Parameters

Parameter	**Type**	**Value Range**
numberOfUEs	Integer	> 0 and $\geq$ *maxNumberOfUEs* in NSI-Rs
minNumberOfLinksPerUE	Integer	> 0 and $\leq$ *maxNumberOfLinksPerUE*
maxNumberOfLinksPerUE	Integer	$\geq$ *minNumberOfLinksPerUE*
numberOfEdgeClouds	Integer	> 0 and $\leq$ *numberOfUEs*
minNumberOfTransLinks	Integer	> 0 and $\leq$ *maxNumberOfTransLinks*
maxNumberOfTransLinks	Integer	$\geq$ *minNumberOfTransLinks* and $\leq$ *numberOfAggClouds*
minCpuEdge	Integer	≥ 0 and $\leq$ *maxCpuEdge*
maxCpuEdge	Integer	$\geq$ *minCpuEdge*
minMemoryEdge	Integer	≥ 0 and $\leq$ *maxMemoryEdge*
maxMemoryEdge	Integer	$\geq$ *minMemoryEdge*
minEdgeReliability	Double	≥ 0 and $\leq$ *maxEdgeReliability*
maxEdgeReliability	Double	$\geq$ *minEdgeReliability*
numberOfAggClouds	Integer	> 0 and $\leq$ *numberOfEdgeClouds*
minNumberOfCoreLinks	Integer	> 0 and $\leq$ *maxNumberOfCoreLinks*
maxNumberOfCoreLinks	Integer	$\geq$ *minNumberOfCoreLinks* and $\leq$ *numberOfMainClouds*
minCpuAgg	Integer	≥ 0 and $\leq$ *maxCpuAgg*
maxCpuAgg	Integer	$\geq$ *minCpuAgg*
minMemoryAgg	Integer	≥ 0 and $\leq$ *maxMemoryAgg*
maxMemoryAgg	Integer	$\geq$ *minMemoryAgg*
minAggReliability	Double	≥ 0 and $\leq$ *maxAggReliability*
maxAggReliability	Double	$\geq$ *minAggReliability*

Parameter	**Type**	**Value Range**
numberOfMainClouds	Integer	$>$ 0 and $\leq$ *numberOfAggClouds*
minCpuMain	Integer	$\geq$ 0 and $\leq$ *maxCpuMain*
maxCpuMain	Integer	$\geq$ *minCpuMain*
minMemoryMain	Integer	$\geq$ 0 and $\leq$ *maxMemoryMain*
maxMemoryMain	Integer	$\geq$ *minMemoryMain*
minMainReliability	Double	$\geq$ 0 and $\leq$ *maxMainReliability*
maxMainReliability	Double	$\geq$ *minMainReliability*
minThroughputRanMean	Integer	$\geq$ 0 and $\leq$ *maxThroughputRanMean*
maxThroughputRanMean	Integer	$\geq$ *minThroughputRanMean*
minThroughputRanStdDev	Double	$\geq$ 0 and $\leq$ *maxThroughputRanStdDev*
maxThroughputRanStdDev	Double	$\geq$ *minThroughputRanStdDev*
minThroughputTransMean	Integer	$\geq$ 0 and $\leq$ *maxThroughputTransMean*
maxThroughputTransMean	Integer	$\geq$ *minThroughputTransMean*
minThroughputTransStdDev	Double	$\geq$ 0 and $\leq$ *maxThroughputTransStdDev*
maxThroughputTransStdDev	Double	$\geq$ *minThroughputTransStdDev*
minThroughputCoreMean	Integer	$\geq$ 0 and $\leq$ *maxThroughputCoreMean*
maxThroughputCoreMean	Integer	$\geq$ *minThroughputCoreMean*
minThroughputCoreStdDev	Double	$\geq$ 0 and $\leq$ *maxThroughputCoreStdDev*
maxThroughputCoreStdDev	Double	$\geq$ *minThroughputCoreStdDev*
minLinkLatencyRan	Integer	$\geq$ 0 and $\leq$ *maxLinkLatencyRan*
maxLinkLatencyRan	Integer	$\geq$ *minLinkLatencyRan*
minLinkLatencyTrans	Integer	$\geq$ 0 and $\leq$ *maxLinkLatencyTrans*
maxLinkLatencyTrans	Integer	$\geq$ *minLinkLatencyTrans*
minLinkLatencyCore	Integer	$\geq$ 0 and $\leq$ *maxLinkLatencyCore*
maxLinkLatencyCore	Integer	$\geq$ *minLinkLatencyCore*
minRanReliability	Double	$\geq$ 0 and $\leq$ *maxRanReliability*
maxRanReliability	Double	$\geq$ *minRanReliability* and $\leq$ 1
minTransReliability	Double	$\geq$ 0 and $\leq$ *maxTransReliability*
maxTransReliability	Double	$\geq$ *minTransReliability* and $\leq$ 1
minCoreReliability	Double	$\geq$ 0 and $\leq$ *maxCoreReliability*
maxCoreReliability	Double	$\geq$ *minCoreReliability* and $\leq$ 1

Table 6.2: Substrate Generator Configuration File Parameters Default Values

Parameter	**Default Value**
numberOfUEs	50
minNumberOfLinksPerUE	1
maxNumberOfLinksPerUE	1

Parameter	Default Value
numberOfEdgeClouds	12
minNumberOfTransLinks	1
maxNumberOfTransLinks	1
minCpuEdge	40
maxCpuEdge	50
minMemoryEdge	40
maxMemoryEdge	50
minEdgeReliability	0.98
maxEdgeReliability	1
numberOfAggClouds	5
minNumberOfCoreLinks	1
maxNumberOfCoreLinks	1
minCpuAgg	50
maxCpuAgg	100
minMemoryAgg	50
maxMemoryAgg	100
minAggReliability	0.98
maxAggReliability	1
numberOfMainClouds	1
minCpuMain	2000
maxCpuMain	2000
minMemoryMain	2000
maxMemoryMain	2000
minMainReliability	0.99
maxMainReliability	1
minThroughputRanMean	50
maxThroughputRanMean	100
minThroughputRanStdDev	10
maxThroughputRanStdDev	20
minThroughputTransMean	100
maxThroughputTransMean	400
minThroughputTransStdDev	10
maxThroughputTransStdDev	40
minThroughputCoreMean	500
maxThroughputCoreMean	1000
minThroughputCoreStdDev	50
maxThroughputCoreStdDev	100
minLinkLatencyRan	1
maxLinkLatencyRan	1
minLinkLatencyTrans	1

Parameter	**Default Value**
maxLinkLatencyTrans	1
minLinkLatencyCore	1
maxLinkLatencyCore	1
minRanReliability	0.98
maxRanReliability	0.99
minTransReliability	0.99
maxTransReliability	1
minCoreReliability	0.99
maxCoreReliability	1

The random substrate generator uses the parameters specified in Table 6.1.
The *numberOfUEs* is the predefined, fixed number of UE groups in the substrate network. It is a positive integer value and must be greater than or equal to the maximum number of UE groups in the NSI-Rs of the generated NSE problem instance.

When generating a random substrate instance based on such a substrate generator configuration file, first the required number of UE groups is created and added to the substrate network.
Then the connections of the UE groups are handled. The generation algorithm iterates through the UE groups and generates a new edge cloud node for every required connection between a UE group and an edge cloud node as long as the defined number of edge cloud nodes is not reached. The associated edge clouds are connected to the UE groups with an edge. The number of the connections between the UEs and the cloud nodes is randomly drawn from the interval

$$[minNumberOfLinksPerUE, maxNumberOfLinksPerUE]$$

using an equal distribution. In this thesis we use $minNumberOfLinksPerUE = 1$ and $maxNumberOfLinksPerUE = 1$, i.e., the range $[1, 1]$. Thus, every UE group is connected to exactly one edge cloud in the generated substrate network. That means, single-connectivity is assumed in this evaluation. This reflects the current structure of mobile networks where the majority of devices and contracts are not capable of multi-connectivity. However, a multi-connectivity scenario can be easily created with the given NSE Test-Case Generator by setting the *minNumberOfLinksPerUE* and the *maxNumberOfLinksPerUE* parameters accordingly.

Figure 6.2 shows the single-connectivity of the UE groups with the edge clouds for a minimal example of a generated substrate network with 8 UE groups ($numberOfUEs = 8$), 3 edge cloud nodes c_0, c_1 and c_2 ($numberOfEdgeClouds = 3$), 2 aggregation clouds c_3 and c_4 ($numberOfAggClouds = 2$) and one main cloud c_5 ($numberOfMainClouds = 1$).
In this minimal example as well as the default configuration, single-connectivity is applied to the cloud node connections. I.e., the following parameters are set to one:

$$minNumberOfTransLink = 1,\ maxNumberOfTransLinks = 1,$$
$$minNumberOfCoreLinks = 1 \text{ and } maxNumberOfCoreLinks = 1.$$

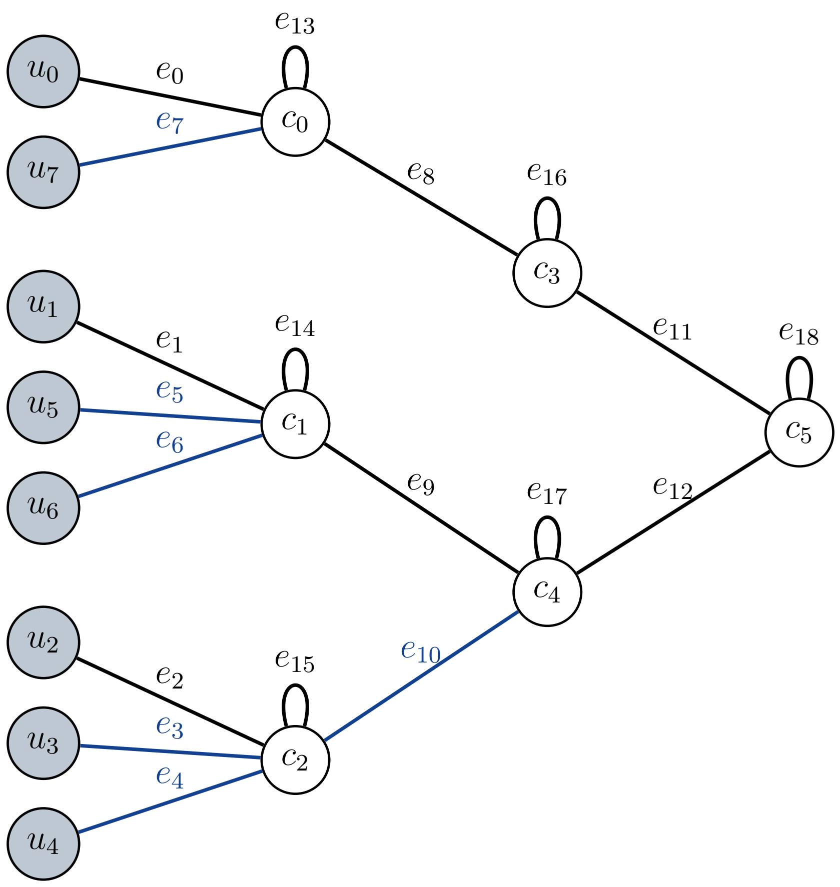


Figure 6.2: Randomly Generated Substrate - Minimal Examle

Some connections are non-deterministic. These randomly defined edges are highlighted in blue color in Figure 6.2. Three edge cloud nodes are created and connected to the first three UE group nodes. The remaining UE groups are connected to random edge clouds. I.e., the UE groups u_3 to u_7 are connected randomly in the minimal example. The aggregation clouds and their connections to the edge clouds are created in the same manner. Two aggregation clouds are created in the example shown in Figure 6.2. The number of connected aggregation clouds is selected randomly from the predefined interval. Due to single-connectivity, the first edge cloud c_1 is connected to the newly created aggregation cloud c_3 and the edge cloud c_1 is connected to the new aggregation cloud c_4. The left-over edge cloud c_2 is randomly connected to one of the two available aggregation clouds. In this example, c_4 is used. Subsequently, all aggregation cloud nodes are connected to the main cloud node c_5. Finally, the free-self-links e_{13} to e_{18} are added to the cloud nodes.
Analogous to this minimal example, the tiny, small, medium and big substrate network instances are generated.

For every cloud node type, the provided CPU and memory capacities as well as the node reliabilities are randomly chosen from a predefined interval with an equal distribution. For each type of cloud node (edge, aggregation and main cloud) specific resource and capability intervals are defined. For instance, the amount of provided CPU resources of one edge cloud is a random (equally distributed) value from the interval $[minCpuEdge, maxCpuEdge]$, while its memory resources are randomly chosen from the interval $[minMemoryEdge, maxMemoryEdge]$ and the reliability of the edge node is between $[minEdgeReliability, maxEdgeReliability]$. The same is applied to the aggregation and main clouds with their specific CPU and memory resource as well as reliability ranges.

The links connecting the UE groups with the edge clouds are referred to as RAN links. The edge to aggregation cloud links are called transport links and the aggregation cloud to main cloud links are named core links. For each type of link a minimum and maximum throughput mean and STD as well as a link latency and link reliability are defined. For example, the RAN links are parametrized with the $minThroughputRanMean$ and the $maxThrouhgputRanMean$. I.e., the mean provided throughput is randomly chosen from the interval

$$[minThroughputRanMean, maxThrouhgputRanMean]$$

with an equal distribution. For NSE without uncertainty, the associated STD, specified by the interval

$$[minThroughputRanStdDev, maxThroughputRanStdDev]$$

is set to $[0, 0]$.
The link reliability of the RAN connections is drawn from the interval

$$[minRanReliability, maxRanReliability]$$

6.1.2.2 Network Slice Instance Requests

The NSI-Rs are randomly created for this evaluation as well. The parameters of the NSI-R Generator Configuration File specify the configuration of the NSI-R Generator. Table 6.3 displays the list of parameters, their type and characteristics. Table 6.4 provides the associated default parameter values.

Table 6.3: NSI-R Generator Configuration File Parameters

Parameter	Type	Value Range
numberOfSlices	Integer	> 0
minSliceWeight	Double	> 0 and $\leq$ *maxSliceWeight*
maxSliceWeight	Double	$\geq$ *minSliceWeight*
minNumberOfAppChains	Integer	> 0 and $\leq$ *maxNumberOfAppChains*
maxNumberOfAppChains	Integer	$\geq$ *minNumberOfAppChains*
minNumberOfAppsPerChain	Integer	> 0 and $\leq$ *maxNumberOfAppsPerChain*
maxNumberOfAppsPerChain	Integer	$\geq$ *minNumberOfAppsPerChain*
minNumberOfUEs	Integer	> 0 and $\leq$ *maxNumberOfUEs*
maxNumberOfUEs	Integer	$\geq$ *minNumberOfUEs* and $\leq$ *numberOfUEs* in Substrate
minNumberOfAppChainsPerUE	Integer	> 0 and $\leq$ *maxNumberOfAppChainsPerUE*
maxNumberOfAppChainsPerUE	Integer	$\geq$ *minNumberOfAppChainsPerUE*
minAppCpuMean	Integer	≥ 0 and $\leq$ *maxAppCpuMean*
maxAppCpuMean	Integer	$\geq$ *minAppCpuMean*
minAppCpuStdDev	Double	≥ 0 and $\leq$ *maxAppCpuStdDev*
maxAppCpuStdDev	Double	$\geq$ *minAppCpuStdDev*
minAppMemoryMean	Integer	≥ 0 and $\leq$ *maxAppMemoryMean*
maxAppMemoryMean	Integer	$\geq$ *minAppMemoryMean*
minAppMemoryStdDev	Double	≥ 0 and $\leq$ *maxAppMemoryStdDev*
maxAppMemoryStdDev	Double	$\geq$ *minAppMemoryStdDev*
minAppReliability	Double	≥ 0 and $\leq$ *maxAppReliability*
maxAppReliability	Double	$\geq$ *minAppReliability* and ≤ 1
minReqThroughputMean	Integer	≥ 0 and $\leq$ *maxReqThroughputMean*
maxReqThroughputMean	Integer	$\geq$ *minReqThroughputMean*
minReqThroughputStdDev	Double	≥ 0 and $\leq$ *maxReqThroughputStdDev*
maxReqThroughputStdDev	Double	$\geq$ *minReqThroughputStdDev*
minReqLatency	Integer	≥ 0 and $\leq$ *maxReqLatency*
maxReqLatency	Integer	$\geq$ *minReqLatency*
minReqLinkReliability	Double	≥ 0 and $\leq$ *maxReqLinkReliability*
maxReqLinkReliability	Double	$\geq$ *minReqLinkReliability* and ≤ 1

Table 6.4: NSI-R Generator Configuration File Parameters Default Values

Parameter	**Default Value**
numberOfSlices	20
minSliceWeight	1
maxSliceWeight	5
minNumberOfAppChains	1
maxNumberOfAppChains	1
minNumberOfAppsPerChain	1
maxNumberOfAppsPerChain	5
minNumberOfUEs	5
maxNumberOfUEs	10
minNumberOfAppChainsPerUE	1
maxNumberOfAppChainsPerUE	1
minAppCpuMean	25
maxAppCpuMean	50
minAppCpuStdDev	5
maxAppCpuStdDev	10
minAppMemoryMean	25
maxAppMemoryMean	50
minAppMemoryStdDev	5
maxAppMemoryStdDev	10
minAppReliability	0.95
maxAppReliability	0.991
minReqThroughputMean	5
maxReqThroughputMean	10
minReqThroughputStdDev	0.5
maxReqThroughputStdDev	1
minReqLatency	5
maxReqLatency	20
minReqLinkReliability	0.95
maxReqLinkReliability	0.991

For every random NSE problem instance a new set of NSI-Rs is created. The number of NSI-Rs is specified by the *numberOfSlices* parameter. The slices weights vary between the *minSliceWeight* and the *maxSliceWeight*, which are randomly selected with an equal distribution from this interval for each NSI-R.

The NSI-Rs are organized in so-called application chains. This is a consecutively connected set of applications. The number of application chains per NSI-R lies in the interval $[minNumberOfAppChains, maxNumberOfAppChains]$. For simplicity, exactly one application chain per NSI-R is used in this evaluation. However, the NSI-R generator provides the possibility to create NSI-Rs with a random number of

application chains. For this evaluation one application chain per NSI-R is sufficient, since more NSI-Rs are used to increase the size and complexity of the problem instances under evaluation, instead of using several application chains per NSI-R.
Each application chain comprises one or several applications connected subsequently. A minimal example of an NSI with one application chain consisting of the applications a_0^0, a_1^0, a_2^0, a_3^0 and a_4^0 is displayed in Figure 6.3.
The number of applications within one application chain is a random integer value between $minNumberOfAppsPerChain$ and $maxNumberOfAppsPerChain$.

The number of UE groups associated with an NSI-R must not exceed the number of UE groups defined in the substrate network, since it is a random subset of the UE groups in the corresponding substrate network.
The number of UE groups assigned to an NSI-R is randomly drawn from the interval $[minNumberOfUEs, maxNumberOfUEs]$.
Every UE group is then connected to a random number of applications chains between $minNumberOfAppChainsPerUE$ and $maxNumberOfAppChainsPerUE$. In this evaluation all UE groups are connected to the only defined application chain.

The required resources and capabilities of the nodes and links of the generated NSI-Rs are based on the resource and capability parameters specified in the NSI-R Generator Configuration File.
Each application is characterized by its required CPU and memory resources as well as its required reliability. If uncertainty in resource demands is considered, then for every node resource a mean and a STD value is specified. In this evaluation the NSI-Rs are randomly generated. The mean and STD values are drawn with an equal distribution from the specified intervals. For instance, the mean application CPU is randomly drawn from the interval $[minAppCpuMean, maxAppCpuMean]$.
For all communication links in the NSI-R the mean and STD of the required throughput as well as the required latency and the required link reliability are randomly selected from the specified intervals. In the certain case, the STD intervals are set to $[0, 0]$ and the mean values are used as the resource demands.

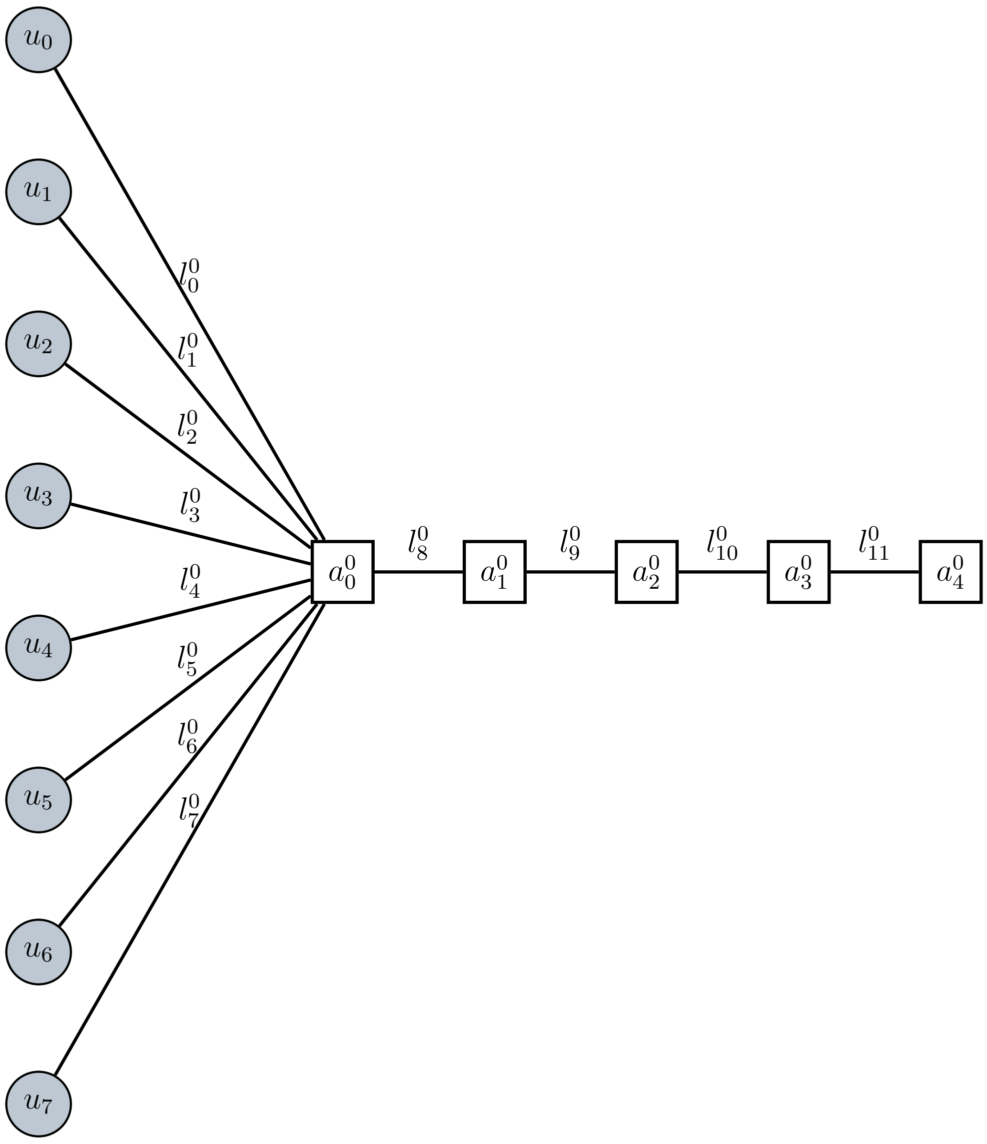


Figure 6.3: Randomly Generated NSI-R - Minimal Example

6.2 Throughput and Path-Splitting Evaluation

This section evaluates the path-splitting variant of the NSE model, introduced in Section 4.3.5.2. The effectiveness of path-splitting is analyzed. Therefore, the acceptance rate, i.e., the percentage of accepted NSI-Rs is analyzed for different ranges of throughput requirements. Additionally, the runtime of the nearly optimal LP-based algorithms are evaluated.

The default configuration introduced in the previous Section 6.1.2 is used as a baseline for this evaluation. However, some modifications are made.
First of all, the *maxNumberOfLinksPerUE* is set to 3 instead of the default value 1. I.e., in the default configuration each UE is connected to exactly 1 edge cloud. In the configuration for the throughput and path-splitting scenario, every UE is connected to between 1 and 3 randomly selected edge clouds. The actual number of links per UE is randomly selected during the generation of the random examples. It is randomly drawn from the set $\{1, 2, 3\}$ with an equal distribution. This resolves the tree structure of the default substrate network and ensures there are several possible paths between the UEs and the connected cloud servers. This is important for path-splitting to show its advantages.
Secondly, the number of UE groups per NSI is reduced to range from 2 to 4 (instead of 5 to 10). This is important since MAI is deactivated in this evaluation scenario. If many UEs are connected to different applications in different NSIs, these applications must be made available to all the, potentially distributed, UEs, i.e., UEs connected with different edge and aggregation cloud servers. Only the main cloud server can serve all UEs concurrently. However, this clashes with the latency restrictions of the virtual communication links on the one hand and the CPU and memory resources of the main cloud server on the other hand. An overall number of 50 UEs in the substrate networks is distributed to 20 NSIs with 2 to 4 UEs each, i.e., 3 UE groups per NSI in average. The generator assures that the UEs are assigned to nearly equal numbers of NSIs. Thus, in average each UE is included in $\frac{3 \cdot 20}{50} = 1.2$ NSIs in this scenario configuration.
Thirdly, the required mean throughputs on the virtual links in the NSIs are modified. Table 6.5 displays an overview of the minimum and maximum required virtual link throughput used in the path-splitting configuration scenarios with low, medium and high throughput requirements.

Table 6.5: NSI-R Path-Splitting Throughput Parameterization

Parameters	**low**	**medium**	**high**
minReqThroughputMean	10	30	80
maxReqThroughputMean	30	80	150

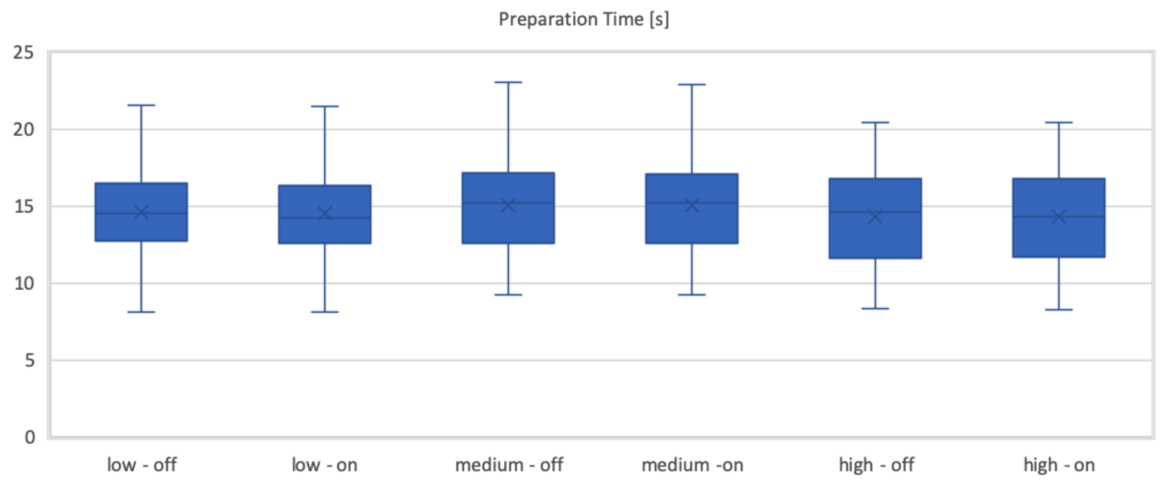
For each set of configurations of the minimum and maximum mean throughput parameters 50 instances of the NSE problem with the default configuration and the modifications explained above are randomly generated for the throughput and path-splitting evaluation. Each configuration is solved with a simple objective function, with MAI and safety-buffers deactivated. The tests are executed once with path-splitting, then the same tests are repeated without the path-splitting option. Figure 6.4 summarized the results. Table 6.6 lists the average acceptance rate as well as the average preparation time and solver time for the evaluated scenarios.

Table 6.6: NSI-R Path-Splitting Evaluation Results

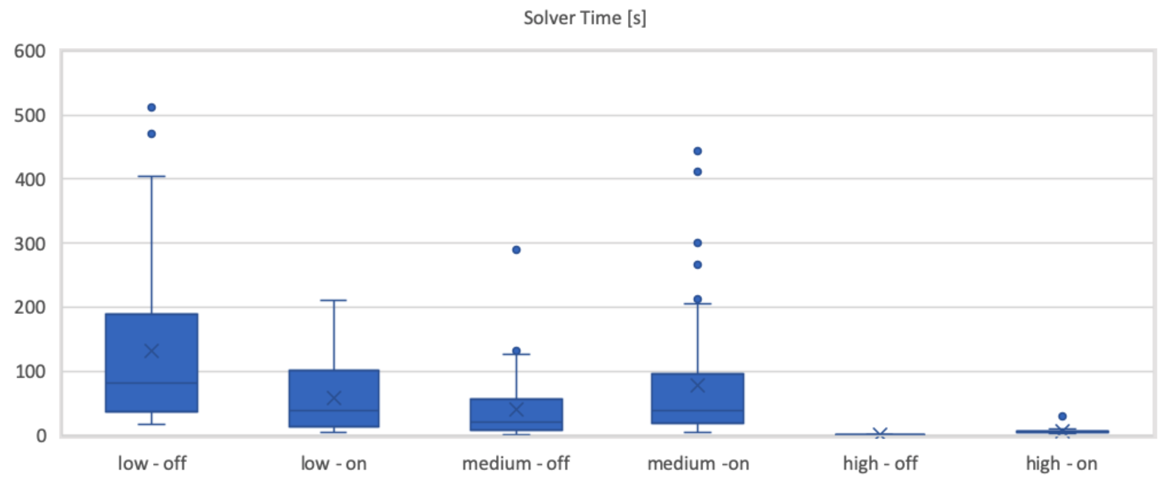
Path-Splitting	**Throuhput**	**Accept. R.**	**Prep. T.**	**Solver T.**
off	low	0.7620	14.6 s	132.6 s
on	low	0.8030	14.5 s	59.1 s
off	medium	0.2460	15.1 s	41.1 s
on	medium	0.3150	15.0 s	78.0 s
off	high	0.0000	14.3 s	0.3 s
on	high	0.0380	14.3 s	6.5 s

As expected, the average preparation time of the NSE scenarios with and without activated path-splitting as wells as for the different intensity of throughput requirements of the links in the NSI-Rs are pretty constant. They only show some noise due to the different randomly created problem instances as well as small fluctuations in computation time. This is expected, since the number of variables and constraints is not affected by the path-splitting option.
The solver time, does not differ much for the different problem instances. For medium and high throughput requirements the solver takes slightly longer when path-splitting is enabled. However, this is not the case for low throughput scenarios.
NSE with path-splitting clearly outperforms NSE without path-splitting in terms of the acceptance rate. For low throughput requirements 76.2% of the NSI-Rs are accepted in average without path-splitting. This improves to 80.3% for the same NSE problem instances, when path-splitting is activated. An even more significant improvement is achieved for the medium and high throughput requirement scenarios. For medium throughput requirements the acceptance rate is improved from an average of 24.6% to an average of 31.5%. In the high throughput scenarios, no NSI-R is feasible when path-splitting is deactivated. Activating path-splitting enables the embedding of at least 0.38% of these extremely demanding NSI-Rs.

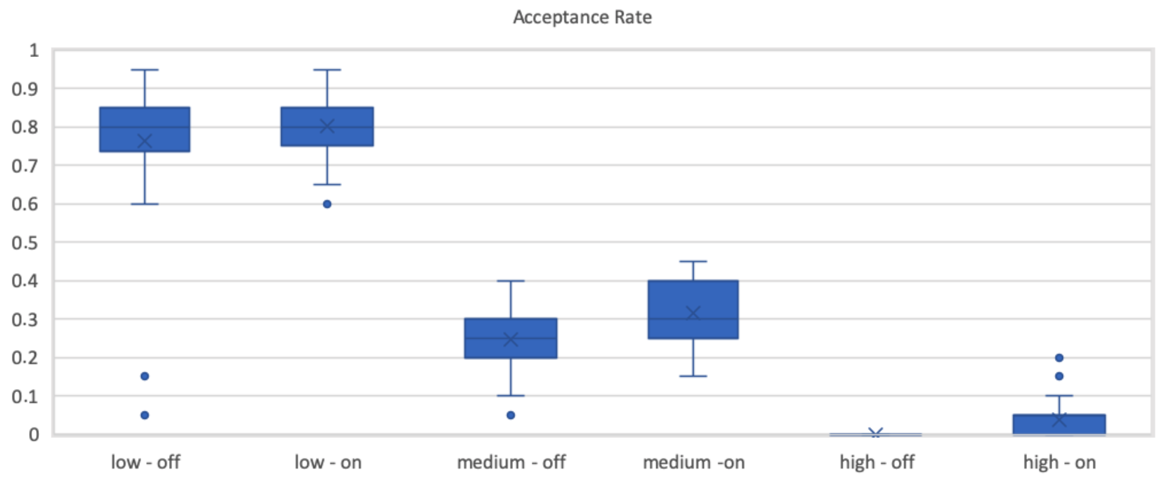
Conclusion To sum up, activating path-splitting when using the simple objective function with MAI and safety-buffers deactivated can improve the acceptance rate, especially if the throughput requirements are high. That means, the share of accepted NSIs is improved, since path-splitting enables new possibilities of embedding



(a) Preparation Time



(b) Solver Time



(c) Acceptance Rate

Figure 6.4: Throughput and Path-Splitting Evaluation

NSIs. Especially for high throughput requirements using multiple physical paths to serve one high throughput virtual path is useful.
If and how much improvement path-splitting achieves in terms of the acceptance rate, is highly dependent on the concrete characteristic of the physical network and the NSI-R requirements. The improvement is higher for higher throughput requirements. This is due to more flexibility in allocating several physical paths to serve one virtual communication link.
Although path-splitting can increase the solver time of the NSE, the impact can be neglected regarding the expected benefits.

6.3 Latency and MAI Evaluation

The latency and MAI evaluation refers to the model described in Section 4.3.5.3. The evaluation shows the effects of MAI on the NSI acceptance rate for different levels of latency requirements. Furthermore, the results are compared with the solution when MAI is deactivated. Additionally, the effect of MAI on the runtime of the NSE algorithm is analyzed.

The evaluation scenario configuration uses the default substrate configuration, as presented in Section 6.1.2. In the NSI configurations only the required latency is modified. Table 6.7 shows the used latency parameters.

Table 6.7: NSI-R MAI Latency Parameterization

Parameters	**very low**	**low**	**medium**	**high**
minReqLatency	1	1	1	5
maxReqLatency	2	5	10	20

For each configuration, 50 randomly generated test-cases based on the default scenario are created. The NSE problems are optimized using the cost and revenue objective function with the activated MAI option compared to deactivated MAI. The results are provided in Figure 6.5 as well as Tables 6.9 and 6.8.

Figure 6.5 (a) presents the number of constraints of the NSE problem instances analyzed in this evaluation. As expected, the number of constraints does not change when MAI is active. Consequently, the preparation time, which is mainly dependent on the number of constraints, is not significantly affected by the MAI option. As Table 6.8 shows, the average preparation time per constraint is relatively constant for the different scenarios.
The solver time decreases with increasing latency requirements, that means, NSI-Rs which require low latency are embedded or rejected more quickly than NSI-Rs with less restrictive latency requirements (high latency).
The acceptance rates indicate that the low latency NSI-Rs are often rejected. This

explains, why they are handled more quickly by the solver. Figure 6.5 (d) shows that NSE with MAI is clearly superior over the basic NSE without MAI for the evaluated scenarios with regard to NSI acceptance rates.
As anticipated, the number of application instances created in average for one NSI application declines with relaxed latency requirements, because it becomes more likely that an application used by several distributed UE groups can be deployed on an aggregation or main cloud node.
The objective values are not evaluated here since, the true revenues and costs are subject to the specific practical scenarios, which are out of the scope of this thesis.

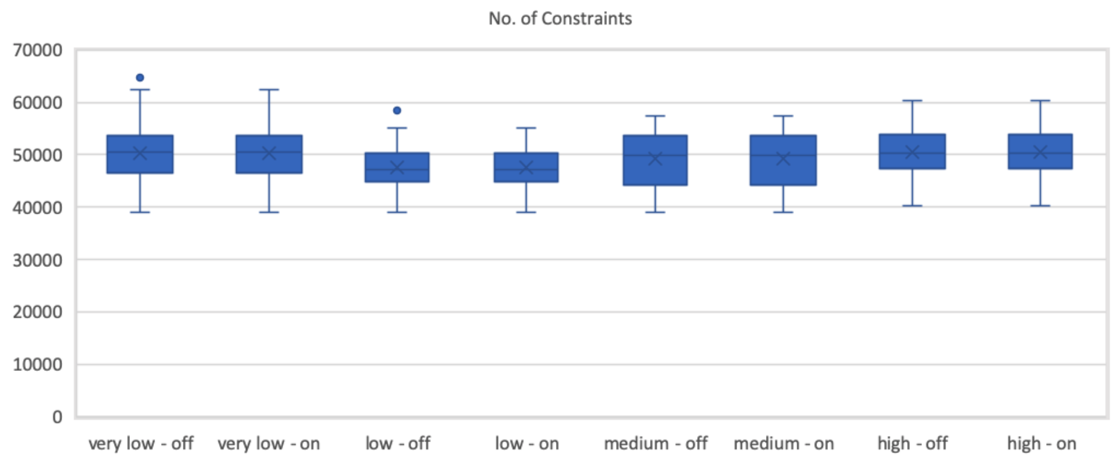
Table 6.8: NSI-R MAI Evaluation Results - Runtime

MAI	**Lat.**	**No. Constr.**	**Prep. T.**	**Prep. T./Constr.**	**Solver T.**
off	very low	50401.5	18.5 s	0.000368	0.3 s
on	very low	50401.5	18.5 s	0.000367	2.7 s
off	low	47624.3	16.7 s	0.000350	0.3 s
on	low	47624.3	16.7 s	0.000351	7.6 s
off	medium	49185.6	17.6 s	0.000357	0.7 s
on	medium	49185.6	17.6 s	0.000358	11.9 s
off	high	50598.7	18.1 s	0.000357	49.3 s
on	high	50598.7	18.2 s	0.000359	37.8 s

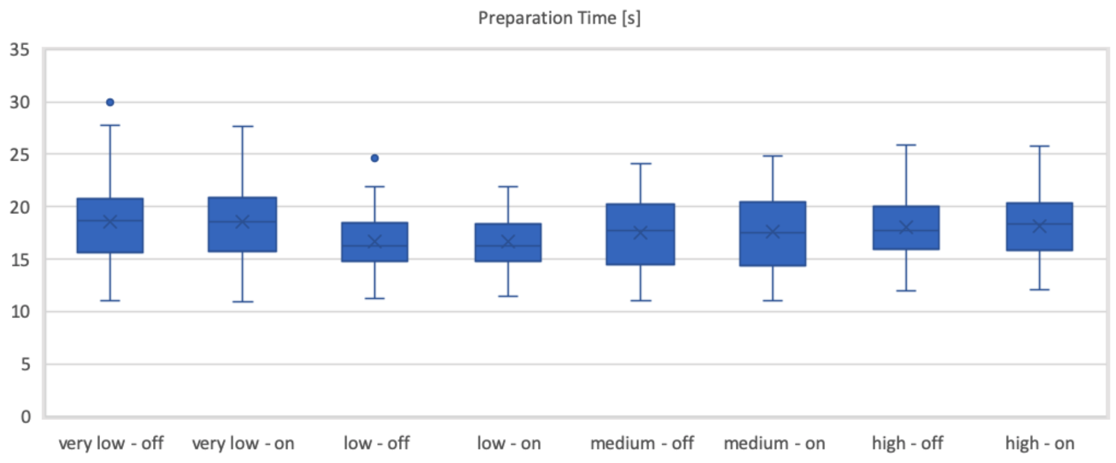
Table 6.9: NSI-R MAI Evaluation Results - Efficiency Improvement

MAI	**Lat.**	**Accept. R.**	**No. App Instances**
off	very low	0.000	1.00
on	very low	0.050	3.53
off	low	0.004	1.00
on	low	0.165	2.08
off	medium	0.093	1.00
on	medium	0.335	1.55
off	high	0.670	1.00
on	high	0.706	1.04

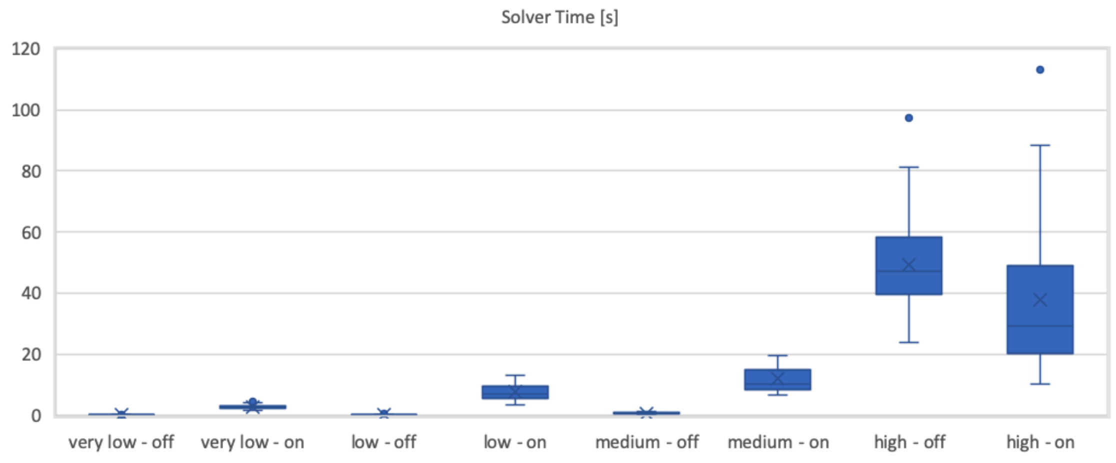
Conclusion In summary, the NSE algorithm with MAI is clearly superior over the basic NSE algorithm without MAI in terms of NSI acceptance rates. The solver time decreases with increasing latency requirements, since NSIs which require low



(a) Number of Constraints

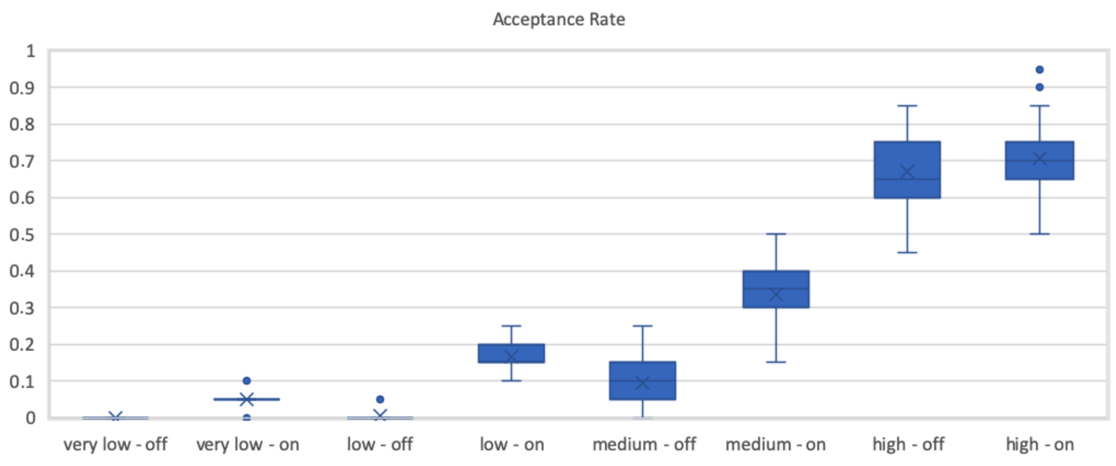


(b) Preparation Time

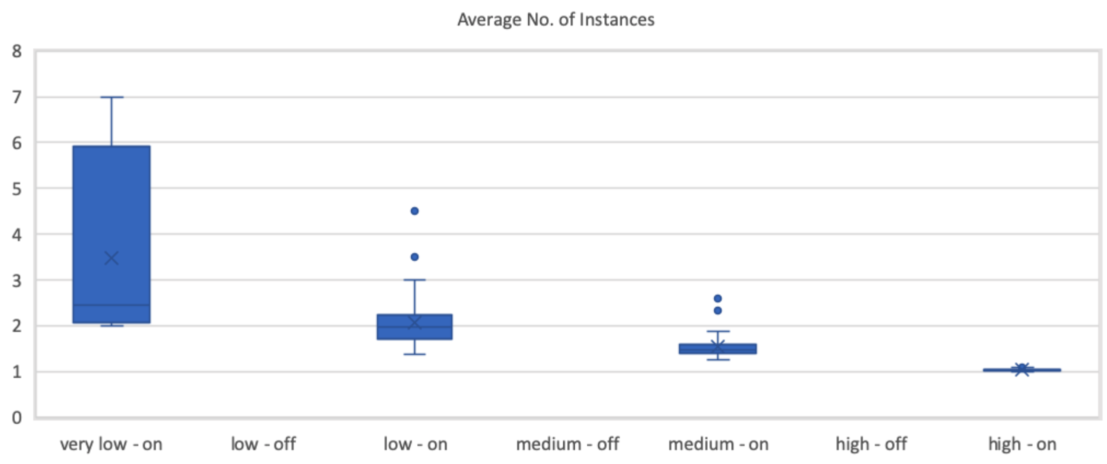


(c) Solver Time

Figure 6.5: Latency and MAI Evaluation 1/2



(d) Acceptance Rate



(e) Average Number of Application Instances

Figure 6.5: Latency and MAI Evaluation 2/2

latency are embedded or rejected more quickly than NSIs with less restrictive latency requirements (high latency). Furthermore, a smaller number of instances of the same application is created in average, when the latency requirements are relaxed.

6.4 Combined Path-Splitting and MAI Evaluation

The combined model using both, the path-splitting as well as the MAI option is evaluated in this section. It is compared with the basic model, without path-splitting and MAI and the models with either only path-splitting or MAI activated. For all versions, the cost and revenue objective function is used. 50 randomly generated problem instances are created. The problem generator uses the default configuration except for the modified parameters listed below.
In order to generate more options for path-splitting the *maxNumberOfLinksPerUE* in the substrate network is increased from 1 in the default configuration to 2 in the configuration for the combined path-splitting and MAI evaluation.
The number of NSI-Rs is increased to 30, since the higher flexibility in embedding, introduced by path-splitting and MAI improves the acceptance rate, a more challenging scenario with more NSI-Rs is used in this evaluation. For throughput, the high throughput configuration of the path-splitting evaluation is used, with *minReqThroughputMean* = 80 and *maxReqThroughputMean* = 150. Challenging latency requirements (between the very low and the low latency scenario) are used, by setting *minReqLatency* = 1 and *maxReqLatency* = 3.

The results of this evaluation are summarized in Figure 6.6 as well as Tables 6.10 and 6.11. The average preparation times of the 50 randomly generated NSE instances are not significantly affected by activating path-splitting, MAI or both options. It is around 59s in average for all model variants. The solver time however, decreases for the path-splitting and combined model. When path-splitting is deactivated and MAI is activated, the solver time shows bigger deviations. It strongly dependents on the concrete example. The combined model provides the most efficient solver times. It reduces from an average of 245.7s for the basic model variant to 88.2s for the combined model.
Naturally, the average number of application instances is 1, if MAI is deactivated. Expectedly, it slightly decreases for the combined model compared to the MAI model.
Considering the objective function values of the cost and revenue objective function for the four different scenarios, the average efficiency increases with every activated feature. This evaluation shows that the combined method is more efficient than the basic, path-splitting and MAI variants. Compared to the basic model variant, path-splitting achieves an average improvement of the cost and revenue objective function value of 1.24%. MAI is even better, with an average improvement of 1.71%. When activating both features, an average improvement in the objective function of 3.98% is reached.
Finally, the deviation of the acceptance rate in percent is analyzed for the four dif-

ferent model variants. The evaluation results show, that the average acceptance rate improves by 1.11% in average, when the path-splitting feature is activated and by 2.14% when the MAI feature is activated, instead. When both features are activated, the acceptance rate improves by 4.84% in average, compared to the basic model variant.

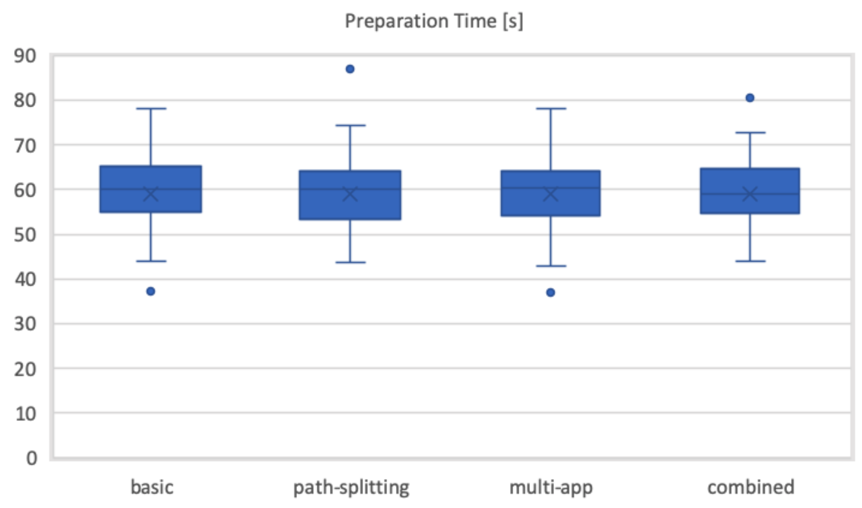
Table 6.10: NSI-R Combined Model Evaluation Results - Runtime

Type	**Prep. T.**	**Solver T.**	**No. App Instances**
basic	59.1 s	245.7 s	1.00
path-splitting	59.2 s	157.6 s	1.00
multi-app	58.9 s	259.2 s	1.05
combined	59.1 s	88.2 s	1.04

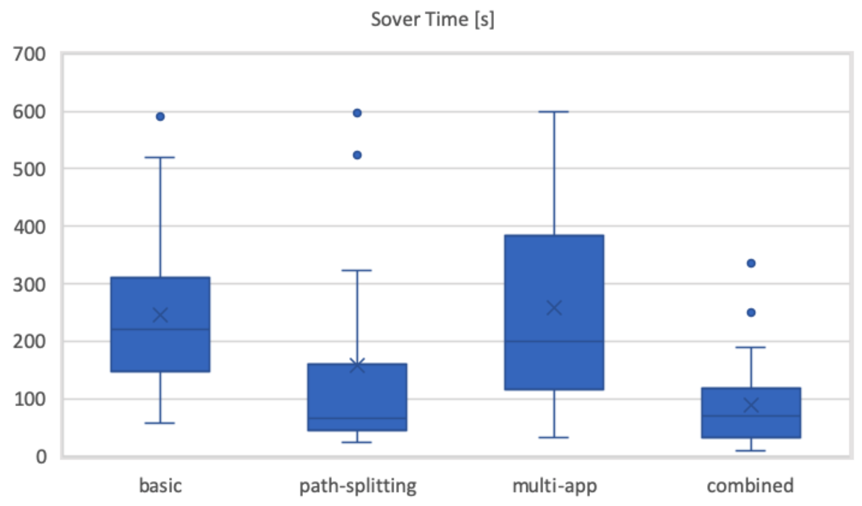
Table 6.11: NSI-R Combined Model Evaluation Results - Efficiency Improvement

Type	**Diff. Objectve F.**	**Diff. Accept. R.**
path-splitting	1.24 %	1.11 %
multi-app	1.71 %	2.14 %
combined	3.98 %	4.84 %

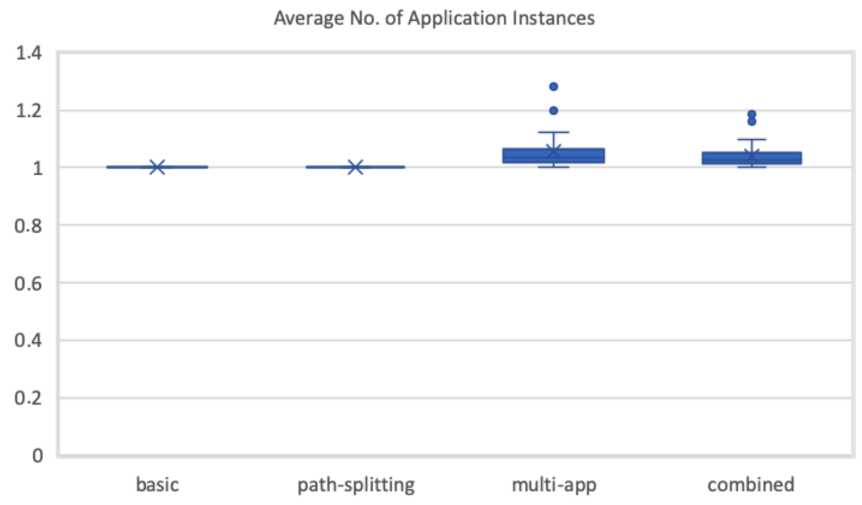
Conclusion The utility of the cost and revenue objective function increases with the activation of the path-splitting and the MAI feature in the NSE algorithm. I.e., more economically beneficial solutions are determined. This goes along with an improved acceptance rate, which increases when the path-splitting, MAI or both features are active. Beyond that, the solver time decreases when using the path-splitting or the combined model.



(a) Preparation Time

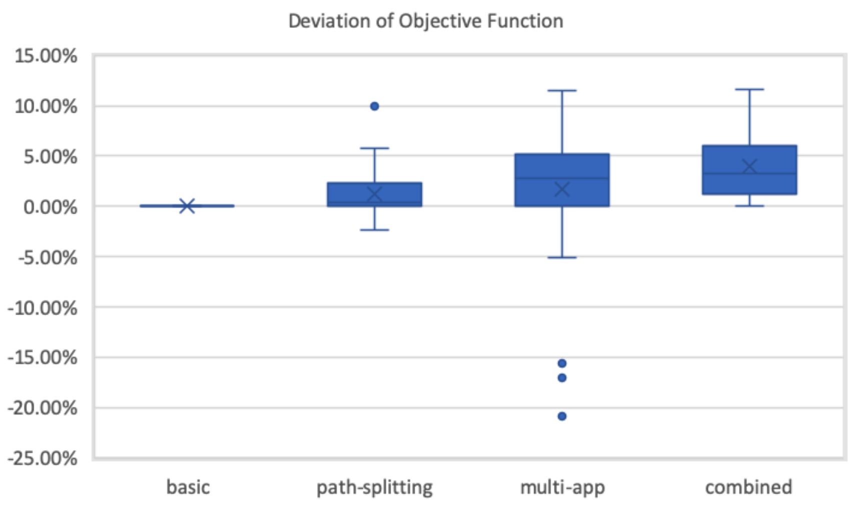


(b) Solver Time

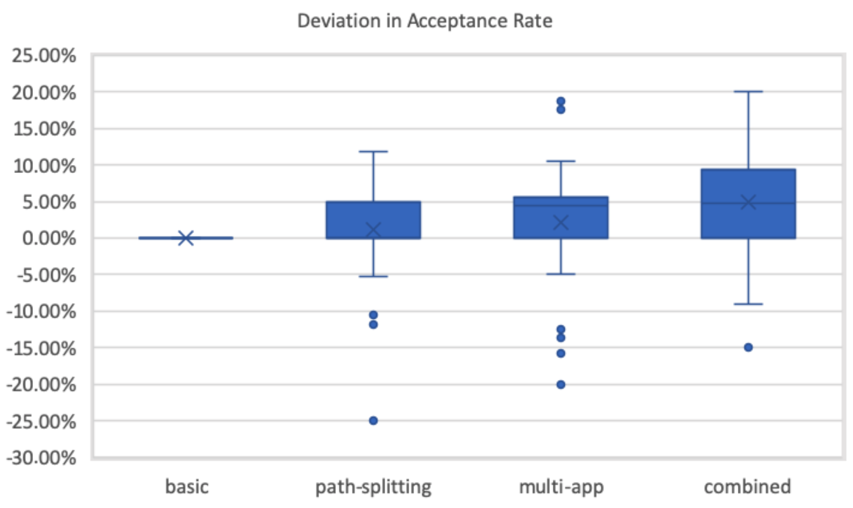


(c) Average Number of Application Instances

Figure 6.6: Combined Model Evaluation 1/2



(d) Deviation of Objective Function Value



(e) Deviation of Acceptance Rate

Figure 6.6: Combined Model Evaluation 2/2

6.5 Runtime and Scalability Evaluation

For this runtime and scalability evaluation of the NSE problem randomly generated substrate networks of different sizes, i.e., with different numbers of elements in the substrate network and the NSIs as well as different numbers of NSIs are analyzed. By default, the simple objective function is used. In all test runs, MAI is active while, path-splitting is deactivated.

Scalability Evaluation Table 6.12 provides an overview over the number of node elements defined for these substrate sizes, the number of NSIs as well as the number of evaluated scenarios for each size.
Due to the long runtimes of the big scenarios, fewer random examples are solved.

Table 6.12: Substrate and NSI Size Parameterization

Parameters	Tiny	Small	Medium	Big
numberOfUEs	20	50	75	100
numberOfEdgeClouds	8	12	16	20
numberOfAggClouds	3	5	5	5
numberOfMainClouds	1	1	1	1
numberOfSlices	10	20	30	40
Nb. of Substrate Nodes	32	68	97	126
Avg. Nb. of Slice Nodes	75	75	75	75
Avg. Nb. of Scenario Nodes	107	143	172	201
Nb. of Scenarios	50	50	50	10

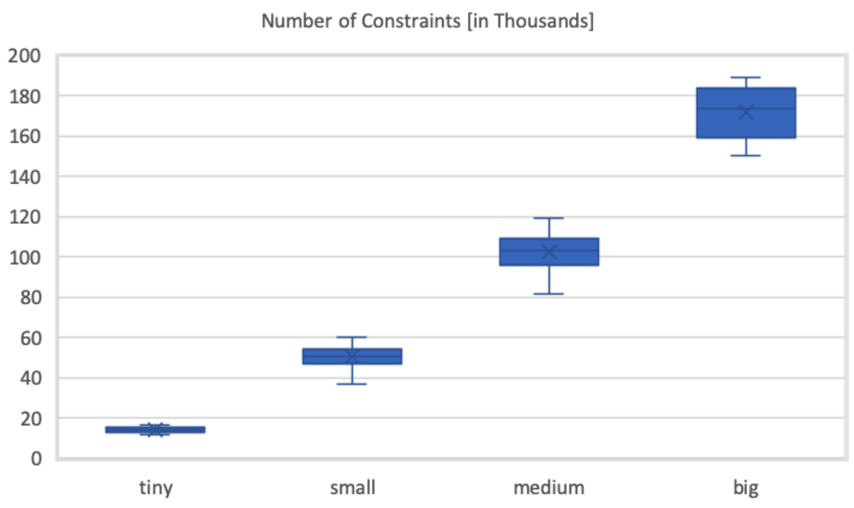
A nearly optimal solution for the $\mathcal{NP}$-hard NSE problem is needed. The configuration of the SCIP solver as well as the particular problem instance influences the accuracy of the solution.
The SCIP solver is configured with a limit for the optimization runtime. The following setting has proven to be a good trade-off between accuracy and calculation time. The so-called *limits/gap* parameter is set to 0.01, that means, when a solution has been found for which the relative gap between the primal and dual solution is below 0.01%, then the optimization is terminated. Otherwise, if 30 branch-and-bound nodes have been processed, since the last improvement of the primal solution hast been seen, the optimization is also terminated. For all other parameters the default configuration of the SCIP solver is used.[53]

Figure 6.7 together with Tables 6.13 and 6.14 provide an overview over the results of the scalability analysis.
When increasing the number of elements in the NSE problem, as defined above for the tiny, small, medium and big scenarios, the average number of constraints increases exponentially, e.g., for an overall number of 32 nodes in the tiny scenarios, the number of constraints is $13,914.16$ in average, while it increases to an average of $171,925.4$ for the big scenarios with an overall average number of 201 scenario nodes. The scenario nodes are the sum of all nodes in the substrate network as well as all NSIs. I.e. the sum of all UE group nodes and cloud nodes in the substrate network as well as the UE group nodes and the application nodes in the NSIs. For simplicity, the communication links are not considered here.
The preparation time also does not scale well, it increases exponentially with the number of nodes in the NSE problem. The preparation time per constraint increases exponentially, this can be explained with the fact that the number of variables, which are part of the constraints, grows exponentially too.
The exponential increase in runtime is most clear when looking at the solver times in Table 6.14. While solving an instance of a tiny scenario only takes a bit more then 3 seconds in average, the solver time of a small scenario is already 10-times more, close to 30 seconds in average. For, medium sized problem instances, the solving time is 10-times more than for the small size, about 333 seconds in average, i.e., nearly 6 minutes. Finally, solving a big scenario takes nearly 27 minutes in average. Thus, only 10 randomly created instances of the big scenario, instead of 50 instances for all other scenarios, where considered in this evaluation.

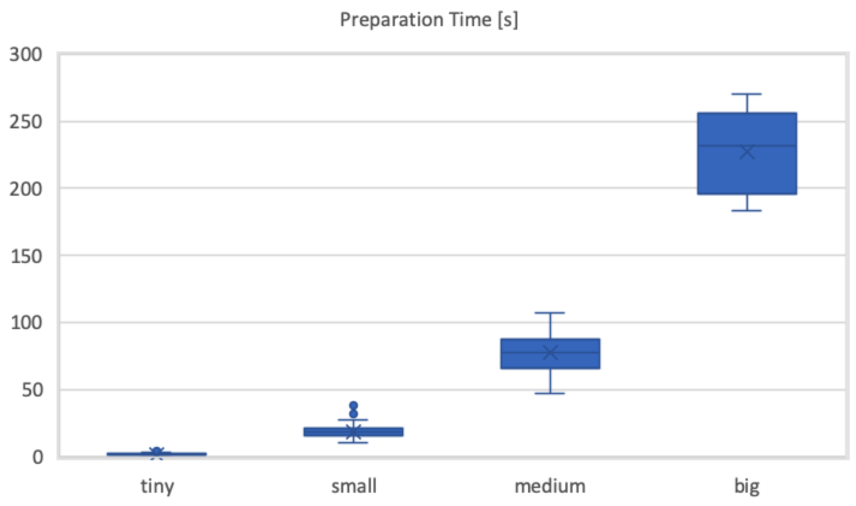
The accuracy of the solutions is measured by the relative duality gap. It lies between an average of 0.174% for the small and 3.7111% for the big scenarios. For linear problems, like the NSE LP formalization, the optimal solution reaches a duality gap of 0%. Although all solutions are close to optimality, the accuracy of the solutions decreases with the size of the problem instances. The decreasing accuracy is due to the solver configuration, since the maximum number of branch-and-bound nodes visited without an improvement of the solution is reached earlier, i.e., at a higher duality gap, for bigger problem instances.

Table 6.13: NSE Preparation Runtime Evaluation

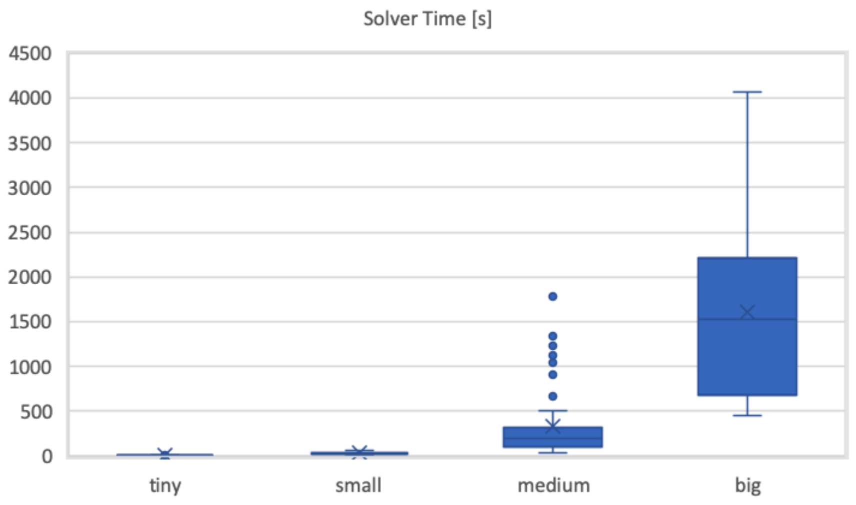
Size	**No. Constr.**	**Prep. T.**	**Prep. T./Constr.**
tiny	13914.16	1.79200 s	0.000128790 s
small	50224.64	18.65158 s	0.000371363 s
medium	102640.88	77.47836 s	0.000754849 s
big	171925.40	227.50730 s	0.001323291 s



(a) Number of Constraints

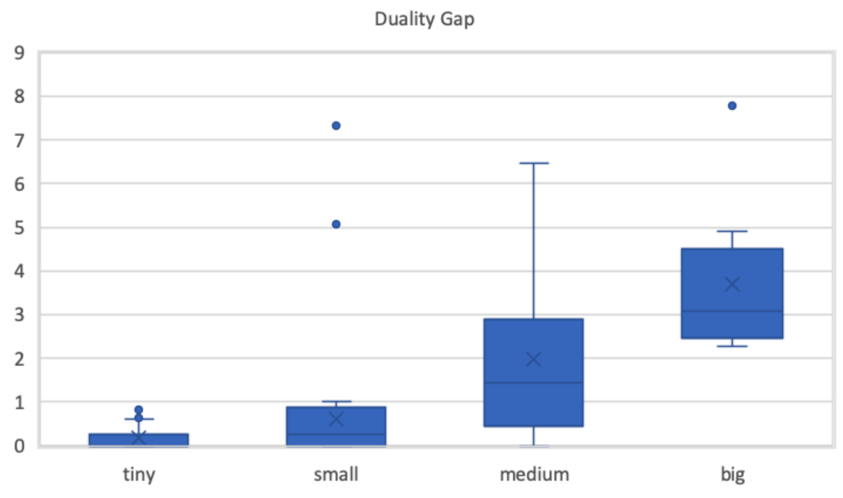


(b) Preparation Time



(c) Solver Time

Figure 6.7: Scalability Evaluation 1/2



(d) Duality Gap

Figure 6.7: Scalability Evaluation 2/2

Table 6.14: NSE Solver Runtime Evaluation

Size	**Solver T.**	**Duality Gap**
tiny	3.08022 s	0.1740 %
small	29.58678 s	0.6176 %
medium	332.66844 s	1.9860 %
big	1601.4383 s	3.7111 %

Objective Function Runtime Evaluation For the small scenarios, the impact of different objective functions, as far as not analyzed in the sections above, is evaluated. Therefore, the 50 randomly generated small scenarios are solved with the simple, latency-aware and cost and revenue objective functions. Then the preparation and the embedding times as well as the acceptance rates are compared relative to each other.

Figure 6.8 and Table 6.15 show the results of the analysis of the runtime and acceptance rates, when the different objective functions introduced in Chapter 4 are used.
Since the preparation time mainly depends on the number of constraints it is not noticeably affected by changing the objective function. However, the average solver time reduces significantly for the latency-aware as well as the cost and revenue objective function compared to the simple objective function. The latency-aware as well as the cost and revenue objective functions practice a much more precise control on the solution of the NSE problem. While the simple objective function only maximizes the sum of embedded NSI weights, the latency-aware as well as cost and revenue objective function additionally perform a fine-tuning of the particular em-

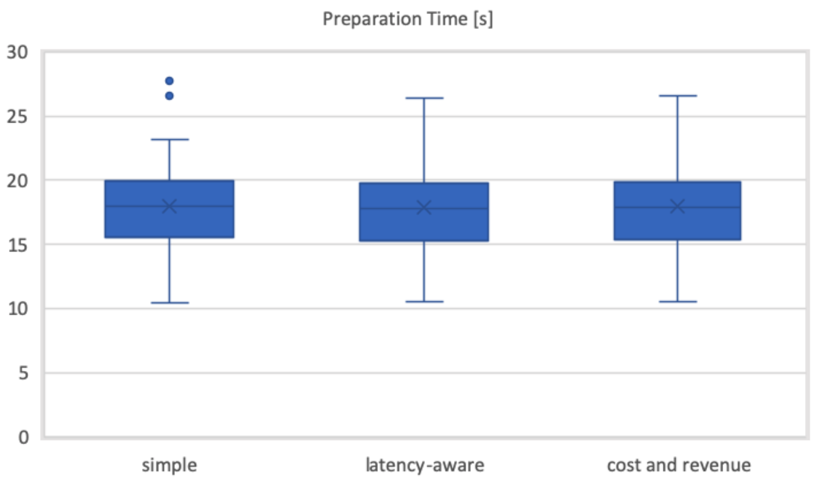
bedding of the elements with the objective of latency or cost minimization. Empiric evaluation shows, that the more complex objective functions improve the runtime with the SCIP solver.

As expected, the acceptance rate cannot be improved significantly since all objective functions have the component of trying to embed as many NSIs as possible. Nevertheless, empirically, the deviation of the quality of the results reduces with the more complex latency-aware as well as cost and revenue objective functions. This leads to the conclusion, that the simple objective function does not offer an advantage for solving the NSE problem, except for being easy to understand. Thus, the more complex objective functions should be chosen for further research as well as for practical applications.

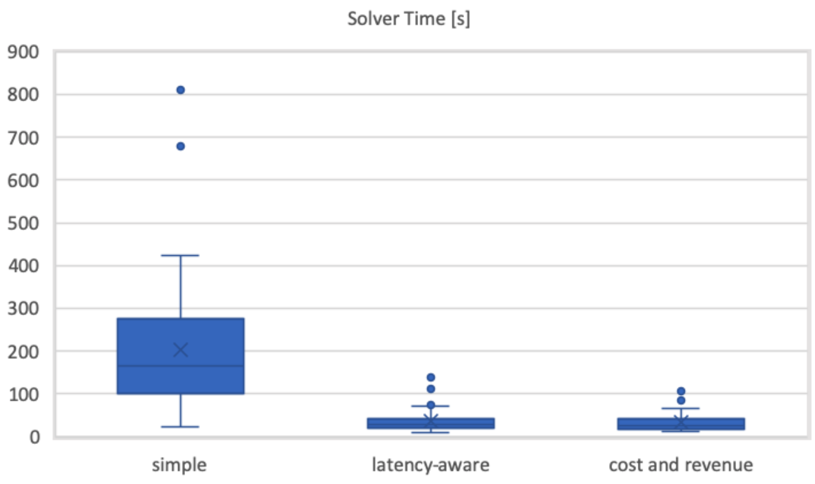
Table 6.15: Objective Function Runtime Evaluation

Objective F.	**Prep. T.**	**Solver T.**	**Accept. R.**
simple	17.9 s	200.7 s	0.685
latency-aware	18 s	35.4 s	0.695
cost and revenue	18 s	33.1 s	0.691

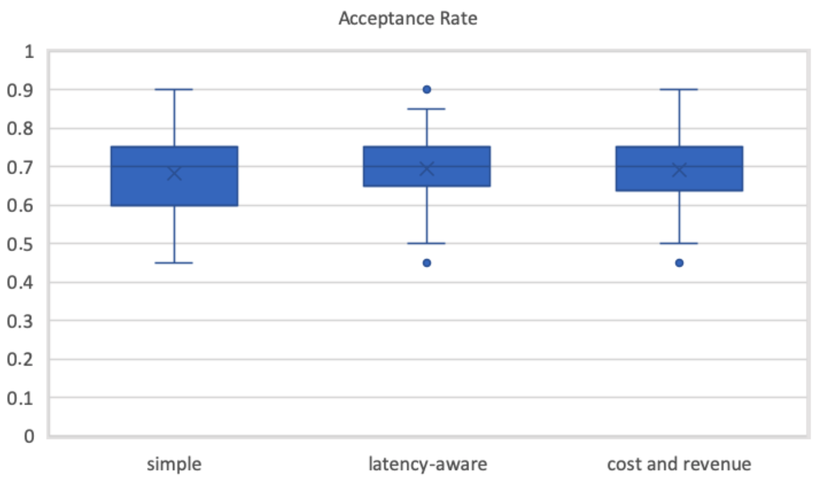
Conclusion The runtime and scalability evaluation shows that the number of constraints and thus the preparation time, which regards the creation and transformation of the variables and constraints, increases exponentially. In addition, the solving times increase exponentially with the number of overall nodes in the scenarios. While the tiny problem instances only take about 3 seconds to be solved in average, the big scenarios are solved in 27 minutes in average. Moreover, the accuracy of the solutions decrease slightly for bigger problem instances.
The comparison of the of the NSE algorithms with different objections functions shows that the acceptance rates are similar for all objective functions, since all have the component of trying to embed as many NSIs as possible. But, the average solver time reduces significantly for the latency-aware as well as the cost and revenue objective function compared to the simple objective function. This is due to much more precise control of the solution.



(a) Preparation Time



(b) Solver Time



(c) Acceptance Rate

Figure 6.8: Objective Function Runtime Evaluation

6.6 Safety Buffer and Robustness Evaluation

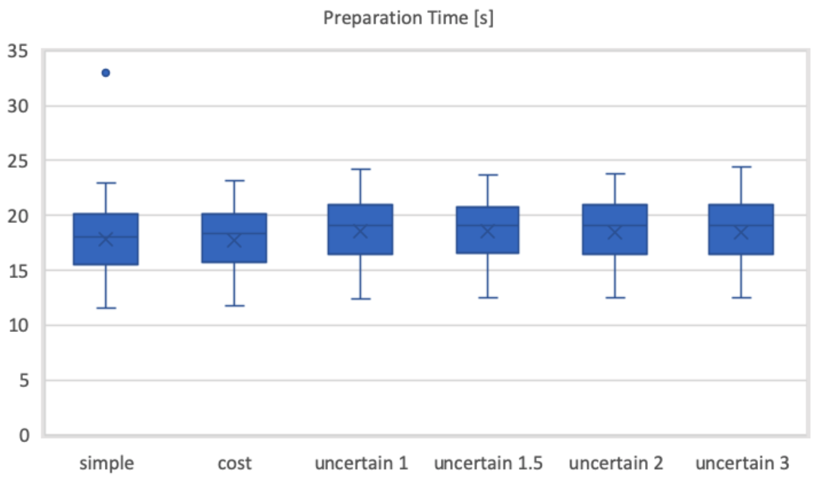
For the robustness evaluation, the default configuration as presented in Section 6.1.2 is used. The model described in Chapter 5 is analyzed for different safety buffers γ. The MAI option is activated, while path-splitting is deactivated for all evaluated NSE scenarios. The 50 randomly generated default NSE scenarios are solved for each value of γ with the uncertainty-aware NSE objective function as well as for the simple objective function and the cost and revenue objective function.
The confidences in resource availability are compared for the different values of γ. Additionally, the runtime of the simple variant and the robust NSE is compared in order to identify the impact of the robustness optimization on the runtime and scalability of the NSE problem.

Table 6.16: Robustness Evaluation Results

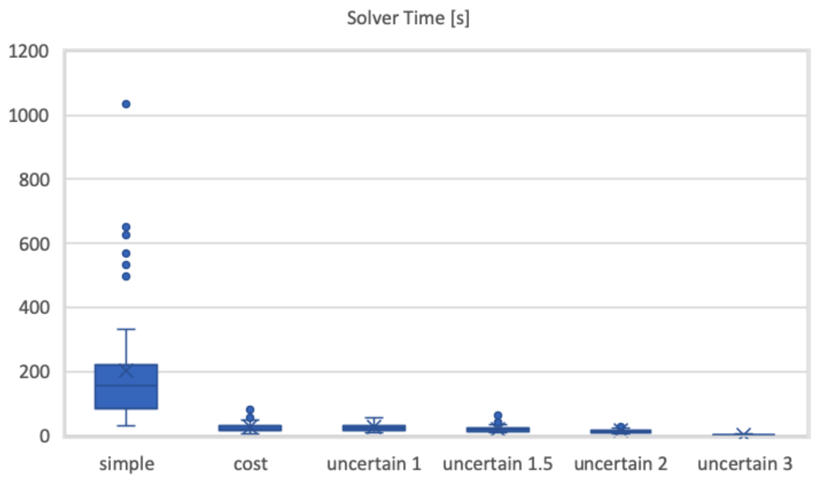
Type	**Prep. T.**	**Solver T.**	**Accept. R.**	**Confidence**
simple	17.8 s	204.4 s	0.692	49.70 %
cost	17.7 s	26.2 s	0.705	63.00 %
uncertain 1	18.5 s	25.3 s	0.519	93.76 %
uncertain 1.5	18.5 s	20.6 s	0.401	97.73 %
uncertain 2	18.5 s	14.7 s	0.284	99.33 %
uncertain 3	18.5 s	1.8 s	0.072	99.96 %

Figure 6.9 as well as Table 6.16 summarize the results of this robustness optimization evaluation.
The average preparation time of the NSE problem is quite similar for the NSE problem under uncertainty compared to the NSE model with MAI.
In contrast, the solver time is reduced by the cost and revenue objective function in comparison with the simple objective function. This is analyzed in the previous Section 6.5. The uncertainty-aware objective function shows similar solver times as the cost and revenue objective function for small values of γ. Increasing the safety buffer γ reduces the amount of usable resources, thus the NSI-Rs acceptance becomes more restrictive. This leads to faster solver times. Overall, empirical analysis shows that NSEs problem instances with stronger restrictions and low acceptance rates can be solved faster with the SCIP solver.

As expected, the acceptance rate is reduced, when robust NSE is used instead of the basic MAI NSE without safety buffers. The higher the safety buffer, the less NSI-Rs can be accepted in average, because additional resources are reserved in order to buffer fluctuations in resource availability and resource demands.
The evaluation shows, that the average NSI confidence of the accepted NSIs is remarkably improved by robust NSE. For the simple objective function and the basic MAI NSE an average NSI confidence of only 49.7% is reached. That means,

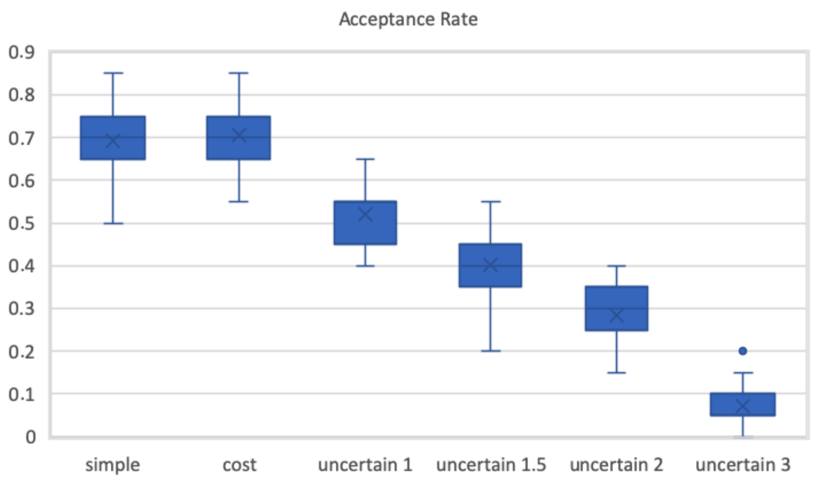


(a) Preparation Time

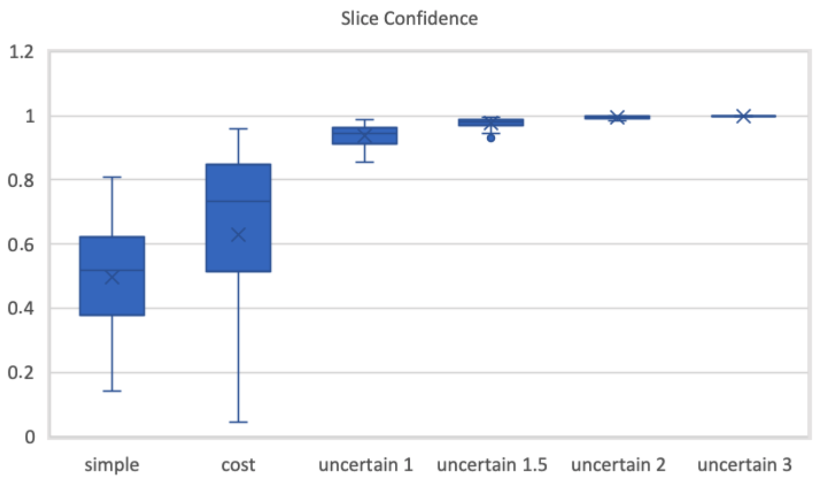


(b) Solver Time

Figure 6.9: Robustness Evaluation 1/2



(c) Acceptance Rate



(d) Slice Confidence

Figure 6.9: Robustness Evaluation 2/2

the probability that all resources of the average NSI are available when required is about 50% for the given probability distributions. When using the cost and revenue objective function this improves to 63%, because in the evaluated scenarios, the elements closer to the core provide more resources and are therefore cheaper on the one hand, but also naturally are more likely to provide a higher resource buffer on the other hand. However, for both, the simple as well as the cost and revenue objective function without safety buffers, the NSI confidence is highly volatile.
When using the robust NSE optimization, the confidence values are dependent on how the safety buffer γ is set. For $\gamma = 1$ the average NSI confidence is already up at 93.76%, but also the STD of the NSI confidences is reduced tremendously, this however comes at the cost of a lower acceptance rate. The acceptance rate declines by 18.6 percentage points, from 70.5% to 51.9%. A further increase of the average NSI confidence is achieved with the higher safety buffers.
Thus, when adjusting γ, a trade-off between the NSI confidence and the acceptance rate has to be made. The configuration of γ depends on the reliability requirements of the NSIs and the available resources or the acceptable costs for overprovisioning.

Conclusion The robust NSE algorithm strongly improves the confidence in resource availability. Reserving a resource buffer which is equal to the STD of the uncertain resources and demands ($\gamma = 1$) already provides an average overall NSI confidence of over 97%. This can be further improved by increasing the safety buffer γ. However, the safety buffers reserve additional resources which are then unavailable for further NSI embeddings. Consequently, the acceptance rate and objective function utilities decline with higher safety buffers and NSI confidences.

6.7 Summary and Discussion

The different variants of the NSE model is analyzed with a dedicated Java implementation, utilizing the SCIP as a solver for LPs. Due to restrictions of the solver interface, the transformation of the object-based representation of an instance of the NSE model is time consuming. Therefore, the runtime efficiency analysis distinguishes between the so-called preparation time of the NSE problem and the solver time. The preparation time comprises setting up the object-based Java constraints as well as the transformation into the LP representation. The solution time regards the pure runtime of the solver.
A default scenario is used and adapted to create different challenges. Usually 50 randomly generated examples are created and solved for each scenario in order to get a clear picture of the behavior of the respective algorithms.
The evaluation analyzes the presented variants of the NSE problem provided in Chapter 4. Especially, the acceptance rate as well as the runtime of the NSE algorithm are considered. Beyond that, the robust NSE, as introduced in Chapter 5, is analyzed. The confidences in resource availability are especially in focus.
Five different evaluation scenarios are analyzed. First, for the evaluation of the

path-splitting model, three different scenarios are used, a scenario for low, medium as well as high throughput requirements of the NSIs. It can be shown, that path-splitting improves the average acceptance rates for all scenarios. Especially in the high-throughput scenario, when the basic NSE model was unable to embed any NSIs in any of the randomly generated instances, with path-splitting it was possible to embed at least a very small percentage of 3.8% of the NSI-Rs.
In the second part of the evaluation, the MAI NSE model is evaluated similarly to the path-splitting model. Four different scenarios, with very low, low, medium and high link latency requirements in the NSI-Rs are used. When MAI is activated for the NSE model the average acceptance rate is massively improved compared to the basic NSE problem.
Thirdly, the runtime, objective functions values and acceptance rates are compared, when path-splitting, MAI or both are activated or deactivated respectively. With each activated feature, the objective function values as well as the acceptance rate improve significantly.
Then, in a dedicated runtime evaluation for the NSE problem four scenarios with different sizes, tiny, small, medium and big are created. The number of elements grows less than linearly for the four different scenarios, nevertheless the number of constraints required in average for the randomly generated examples for each category increases exponentially. The preparation and solver times also increase exponentially. This leads to the conclusion that further measures are required for solving large instances of the NSE problem in its different variants with a short runtime.
As a part of the scalability analysis, the accuracy of the solutions is evaluated. The results show that the NSE solutions are close to optimality.
The different objective functions of the NSE problem are compared to each other. The solver time for the latency-aware and cost and revenue objective function compared to the simple objective function are significantly better in average, while the average preparation time as well as the average acceptance rates remain quite constant.
Finally, the evaluation of the robust NSE model shows that the NSI confidence can be significantly improved. Beyond that, the parameter γ provides an effective tool to manage the trade-off between the confidence in resource availability and the resource utilization. The resource utilization is important for the costs of an NSI as well as the acceptance rate of the overall NSE problem.

7 Related Work

This chapter provides an overview and discussion of the related work of this thesis. In the first section, an overview over literature on VNE and NSE is given and discussed. The second section summarizes and discusses recent scientific literature on NSE in conjunction with Edge Computing. The third section especially focuses on research on VNE under uncertainty, resource overbooking and robust algorithms for the VNE problem under uncertainty. The chapter is concluded by a summary and a consideration of the similarities and differences between the NSIA approach presented in this thesis and the related work in literature.

7.1 Virtual Network Embedding

The NSE problem is a special case of the well-researched VNE problem. The VNE problem belongs to the class of $\mathcal{NP}$-hard problems, first proved in [28], see also [30]. It is a well-researched and understood mathematical problem. In literature, various algorithms and heuristics for the VNE exist, including optimal solutions using ILP. The VNE problem is getting increased attention in practice. Amongst others, the telecommunication industry is adopting network virtualization techniques. However, there is a lack of concepts for embedding virtual end-to-end networks onto a common mixed wired and wireless physical network.
Fixed network embedding is very well-researched. Fischer et al. provided a survey on VNE, focusing on fixed networks in 2013, see [27]. But the advancements in VNE are also evaluated for wireless networks, see for example the papers of Riggio et al. [54] and Tsompanidis et al. [55]. Moreover, Richard et al. [4] published a survey on recent work in mobile network slice embedding. For example, Esposito et al. propose a distributed approach for the $\mathcal{NP}$-hard network slice resource allocation problem in form of a consensus-based auction mechanism [56]. Two possible policy configurations are analyzed. In the Single Allocation Distributed slice embedding only one bid on one virtual node can be made per auction round. In contrast to that, the Multiple Allocation Distributed slice embedding allows to bid on several virtual nodes simultaneously. While Multiple Allocation Distributed has a lower convergence time, Single Allocation Distributed results in a better load balancing and is faster in deciding on the feasibility of a network slice embedding. The virtual links are embedded in the next step, after the virtual nodes have been assigned completely. Yang et al. [57] use a karnaugh-map based heuristic for an efficient virtual network embedding. In a first step, the wireless resources are divided by, e.g.,

TDMA or FDMA into resource blocks. In the second step, the virtual networks are assigned to the resource blocks using a karnaugh-map based concept. The authors show that this is a feasible and efficient way of wireless virtual network embedding. However, in both approaches the actual available throughput provided to a network slice, depending on varying channel conditions, as well as important capabilities, like latency and accessibility are not considered. Moreover, they do not allow to analyze possible resource overbookings and assume stable resource provisioning and utilization. These approaches have in common that they deal with network slice resource allocation at runtime, i.e., focus on resource partitioning and sharing as well as on network slice isolation. Network slice isolation refers to the problem of assuring that the slice specification is not violated because of changes in another slice. The authors of [4] criticize that the evaluated algorithms are described very vague and that they are based on assigning a certain number of Physical Resource Blocks (PRBs). This way no performance guarantees can be given. Varying chancel conditions make it hard to determine a suitable number of PRBs. Richard et al. [4] state that higher-level variables, such as a percentage of the total resources, should be used instead of PRBs. However, this comes at the cost of hardly being able to give guarantees on the amount of resources provided to the owner of the network slice.
In contrast to these activities, the model proposed in this thesis does not cover network slice deployment and issues like network slice isolation and PRBs assignment during runtime. Taking the results of Richart et al. into account, this work abstracts from PRBs and directly draws on network performance parameters, like throughput and latency.

Zhang et al. [58] present an approach for runtime optimization of throughput resources for NSEs. Wang et al. [59] are aiming at resource price balancing, considering dynamic offer and demand of NSI resources and Jiang et al. [60] use UE admission control to avoid network overloading at runtime. Vassilaras et al. summarize the algorithmic challenges of efficient NSE in their paper [61]. They propose an ILP model for finding an optimal solution for a simplified network slice embedding problem and state that the solution with ILP would take tens of minutes. In order to achieve faster run times, an efficient heuristic targeting at a nearly optimal solution is required. However, developing such a heuristic remains an open research question. In addition to that, Vassilaras et al. mainly focus on problems related to network slicing at run time, for instance, the authors shed light on the challenges regarding end-to-end latency constraints, heterogeneous networks, multi-tenancy and slice fairness as well as the issue of dynamic network slicing and online optimization. **NSI deployment and resource management at runtime are not covered by this thesis. This work focuses on finding a nearly optimal utilization of the NSI resources in a mixed wired and wireless end-to-end network during the NSI preparation phase.**

Despotovic et al. [31] propose a scalable algorithm for solving the VNE problem optimally and efficiently. Their problem formulation, the so-called VNetMapper,

allows to solve a VNE with hundreds of nodes and thousands of links within a few seconds. However, the VNetMapper is not optimized for end-to-end mobile networks. It only considers consumable resources, e.g., throughput and memory. Moreover, it does not apply for end-to-end mobile NSIs, since it does not consider network quality parameters, like link latency, availability and reliability. Constraints for such non-consumable capabilities are not formalized by the approach. Beyond that, the VNetMapper, proposed in [31] is only capable of a one-to-one mapping of virtual to physical nodes and links. That means, it is not possible to map several virtual nodes or links on one physical node or link of the substrate. This is not a feasible constraint for the NSE problem, since the network infrastructure, e.g., antennas, base stations and transport links, must be shared among several NSIs.
In contrast to that, this thesis presents an approach that analyses the available resources of mixed wired and wireless end-to-end mobile networks and provides a nearly optimal embedding of mobile NSIs into this shared physical network.
Usually, ILP solutions for VNE only perform reasonably well on small networks. However, NSE is associated with large problem instance, numerous parameters and constraints. Thus, the approach of this thesis uses the design guidelines for Integer Programming proposed by Despotovic et al. [31] for efficient and scalable, nearly optimal VNE models. The model proposed in this thesis is tailored to end-to-end mobile network slicing. It takes latency and reliability constraints into account and optimizes the latency of the embedded NSIs. Furthermore, it is capable of many-to-one mappings and the embedding of network functions chains and takes advantage of the fact that the UE nodes are the same in the physical as well as the virtual networks.

7.2 Network Slice Embedding and Edge Computing

The challenges of NSE in conjunction with edge computing are currently under intense discussion. One related publication is the paper of Sanguanpuak et al. [62]. The authors model the scenario of network slice allocation to multiple MNOs. The allocation problem is solved with the generic Markov Chain Monte Carlo Method. The method considers latency constraints and targets at minimizing infrastructure costs. The optimal mapping solution is calculated with the greedy fractional knapsack algorithm. This algorithm focuses on determining the best network slice to MNO resource mapping for co-located resources.
Xian et al. present a Mixed Integer Nonlinear Problem formulation of resource allocation with edge computing in [63]. The proposed optimization algorithm takes the link and server capacities as well as latency restrictions into account. Sequential Fixing is used to efficiently compute nearly optimal solutions for the mobile edge resource allocation problem.
In contrast to the publications [62] and [63], this thesis provides an end-to-end NSE solution with distributed UEs using a common network slice.

The number of required instances of a specific application within a network slice is determined automatically, depending on the topology of the network, the end-to-end latency, throughput as well as CPU and memory requirements of the respective application. The approaches mentioned above focus on choosing the optimal edge cloud for a predefined application instance associated with a specific UE group at a specific location. However, they do not provide an automated optimization of MAI for geographically distributed UEs in consideration of latency requirements.

Another publication in this area is the work of Song et al. [64]. The authors of this paper propose a method of allocating VNFs in slicing-enabled 5G networks using an edge computing infrastructure in order to reduce network service latencies. Therefore, the required number of instances of the VNFs are determined. The approach also integrates network function chaining.
But, in comparison to this thesis, the presented approach is not integrated with solving the NSE problem.

Further approaches focus on the allocation of computation tasks on distributed clouds during runtime. For example, Alicherry et al. [65] present an efficient approximation algorithm for allocating computation tasks on a distributed cloud with the objective of minimizing communication costs and latencies. Hao et al. [66] provide online heuristics for the resource allocation problem for geographically diversified cloud servers. Both papers are providing algorithms for solving the NSE problem. They are both taking latency and throughput restrictions as well as the network topology into account. However, they are not tackling the challenges of edge computing in conjunction with the NSE problem. Beyond that, for instance, Akhatar et al. [67] present an ILP-based solution for the virtual function placement and traffic steering in 5G networks under the assumption, that every service instance is deployed exactly once.
In contrast to that, the algorithm presented in this paper optimizes the number of application instances and their placement in the physical network. Thereby, throughput and latency on the links as well as the network topology and the NSE problem are considered. To the best of the authors knowledge, none of the approaches in literature is tailored to end-to-end NSE leveraging MAI and edge computing at the same time.

7.3 Uncertainty Evaluation and Overbooking

Various approaches on assigning and overbooking network resources can be found in literature. In [68] the authors show that careful network overbooking can save costs for the user, while concurrently increasing the revenue of the network service provider. In this paper, a game theoretical approach is used. Like most publication on virtual network overbooking (see for instance [69, 70, 71, 72, 73, 74]) it

only focuses on the overbooking of a single resource without considering potential interdependent other resources. **That means, these publications do not provide a comprehensive solution for the VNE and overbooking problem as it is proposed in this thesis.** For example, Fiedler [69] provides an approach for careful virtual network overbooking focusing on the availability of one resource shared by several users. It is based on a per user embedding with an artificial traffic model. **This concept cannot be directly transferred to NSE in end-to-end mobile networks. In contrast to this thesis, it also does not include the optimization of the resource allocation.**
Several publications, for instance, Ball et al. in [75] and Liu et al. in [76] propose an optimal communication link overbooking ratio calculation for telecommunication networks maintaining a predefined Quality of Service (QoS) level. Sadreddini et al. [77] provide a framework for cognitive radio networks for finding the optimal compensation rate for network overbooking using Particle Swarm Optimization.
In contrast, this thesis aims to embed as many NSIs as feasible, while allowing a careful overbooking of several partly interdependent resources, combined with numerous further qualitative and quantitative feasibility constraints. Furthermore, the confidence in resource availability for the elements, resources and NSIs are determined.

In addition, several promising approaches on assigning virtual network resources, allowing overbooking, can be found in literature.
In [78], Marotta and Kassler present a robust VNF placement algorithm which is based on Γ-robust optimization to protect the solution against data uncertainty. Γ-robust optimization is a special case of robust optimization. Robust optimization is linear optimization under uncertainty. In contrast to stochastic optimization, robust optimization is not based on a known probability distribution, instead it optimizes the objective function over a specific parameter space, the so-called uncertainty set. Γ-robust optimization is used when the objective function parameter as well as the right side of the constraints, e.g., the amount of provided resources, are assumed to be certain and a predefined number Γ of the uncertain parameters can deviate from their nominal value. The Γ-robustness optimization models are LPs. Thus, the proposed model provides an efficient node and link mapping for the optimization of the energy efficiency on an adjustable protection level.
Beyond that, Marotta et al. propose an even faster three-step heuristic, which also considers latency constraints, in [79].
In this thesis, a different effective approach to achieve robust solutions of the NSE problem under uncertainty with an LP is presented.

Blanco et al. [80] present a robust VNF placement optimization model for optimizing the power consumption in 5G mobile networks which mitigates service demand uncertainty, while Altın et al. [81] provide a robust VNE algorithm considering communication traffic patterns. Chochlidakis et al. [82] focus on an adjustable tradeoff between robustness and resource utilization of the embedding, taking user mobility into account. Reddy and Baumgartner et al. propose a similar approach in [83]. In

order to increase profits by network overbooking while providing a high probability of feasibility they use Γ-robustness optimization with a mixed integer linear program to handle, for instance, unpredictable short-term mobile traffic increase, e.g., flash crowd events, see [84].
Coniglio et al. provide an exact as well as a heuristic solution for the offline version of the VNE problem based on MILP and Γ-robustness optimization in [85]. Building on this, they developed a chance-constraint formalization of the general VNE problem without latency constraints in [86].
These approaches have in common, that the probability distributions of the uncertain parameters are unknown. Thus, they need to protect the solution against possible variations within a predefined uncertainty budget. However, this leads to a less beneficial solution, since the objective of resource efficiency has to be balanced with the protection against resource uncertainty.
This thesis, however, assumes that the data history on mobile network resource availability as well as resource utilization of the deployed and running NSIs is available. The data can be used to determine an estimated probability distribution for the resource availability and resource utilization. For new NSIs, that have not been deployed before, an estimation of the resource requirements can be made based on the SLA requirements, the type of network slice and the resource utilization data of similar, already deployed NSIs.

The work of Trinh et al. [87] proposes an overbooking mechanism for virtual networks. Their work is based on soft-guaranteed service levels providing a percentage of time with full bandwidth or service availability and a reduction factor for the limited availability. The main focus is on calculating the price reduction which can be offered per user to network slice tenants that are willing to accept a predefined limitation of service quality for a single resource, like bandwidth.
This thesis pursuits the opposite approach, for given NSI requirements with specific resource and capability demands the best NSE is calculated and the risk of violating the SLAs of the NSIs is determined for this embedding.

7.4 Summary and Discussion

There is little literature on models and algorithms for feasibility and robustness analysis in the NSI preparation phase. However, when expanding the scope of the related work, one can find a lot of publications on solving the general VNE problem as well as many publications on resource allocation for network slicing during runtime. But, most VNE algorithms focus on fixed networks. Beyond that, usually assumptions are made which are unsuitable for the NSE problem in 5G mobile networks. For instance, the one-to-one mapping is a very typical assumption for the VNE problem. That means, only one virtual node can be mapped on a physical

node and every virtual node can only be mapped once. Furthermore, UEs with their specific characteristic in the embedding model are usually not considered.
When it comes to publications on NSE, there is a lot of related work on NSI allocation during runtime. However, little research on the NSI feasibility analysis and resource allocation in the preparation phase can be found in literature. Nevertheless, the NSI feasibility analysis and resource allocation in the preparation phase is crucial in order to enable quick NSI acceptance via an automated 5G Network Slice Customer Portal.

Several approaches towards resource allocation in mobile networks with edge computing have been published. However, prior art lacks a holistic solution for NSE in conjunction with MAI. None of the related publications considers the allocation of services on distributed clouds and solves it together with the NSE problem. A comprehensive consideration of cloud computing with MAI and NSI-R feasibility analysis is to the best of the authors knowledge not part of previous literature.

Regarding NSE under uncertainty, various related approaches can be found in literature. Some of them focus on determining the cost-optimal network resource overbooking level or the compensation rate for a single resource. Moreover, several approaches on assigning virtual network resources to potentially overbooked physical resources are presented in the prior art. These solutions are based on other concepts than the ILP-based comprehensive robust embedding LP approach provided in this thesis. Most solutions either focus on VNF placement only or consider the general VNE problem without latency constraints and UEs. In addition, the approaches provided in literature usually assume that the probability distributions of the uncertain parameters are unknown. However, this thesis is based on historical data on resource utilization of already running NSIs and estimations for new NSI-Rs. This data is used to determine the probability distributions of the parameters. To the best of the authors knowledge, none of concepts in literature considers the provisioning of the confidence in resource availability and the risk of violation of SLAs for network slicing in end-to-end mobile networks, taking the RAN, fixed networks and cloud server resources into account.

8 Conclusion and Outlook

In the previous chapters, the NSIA process standardization and models have been presented. Moreover, the prototypical implementation allows a comprehensive evaluation of the different model variants and objective functions as well as the robust NSE.
In this chapter, this work is summarized and concluded. Lastly, some ideas on future improvements and extension of the NSIA algorithms are briefly presented.

8.1 Conclusion

NSaaS in 5G mobile networks requires reliable and runtime- as well as resource-efficient NSE algorithms to enable a quick network slice feasibility analysis and on-demand resource allocation via a customer portal. This thesis provides an NSIA process description. The NSIA process allows to decide whether to accept or to reject an incoming NSI-R taking the individual business policies and risk tolerance of the MSP into account. In addition, the corresponding NSIA and subordinated feasibility checking services in the context of the ETSI ZSM Reference Architecture Framework are developed. Through this thesis, network resource overbooking as a necessity for profitable network operation in an uncertain environment is considered.

Based on the NSIA process and the feasibility checking service standardization, a formal model of the NSE problem is presented in this work. The model allows to solve the NSE problem nearly optimally using an out-of-the-box LP solver. The mathematical NSE model has a modular structure and can be executed in several different variants. These variants enable, many-to-one mappings, network function chaining, path-splitting and MAI as well as different objective functions. With the ability to realize many-to-one mappings, that means, mapping several virtual applications or network functions, belonging to the same or different NSIs, on the same physical element (server or cloud node) and the possibility of defining and embedding NSIs with network function chains, the proposed NSE algorithm is suitable to perform the resource feasibility check as well as the NSIA and the embedding process for NSaaS.
Optionally, path-splitting and MAI can be activated. Path-splitting enables more flexibility for the virtual link embedding, since it allows to combine several physical communication paths to serve one virtual communication link.
MAI particularly enables processing low-latency requirements in conjunction with

edge computing. Using the build-in optimization, the presented algorithm determines the nearly optimal number of application instances required to serve distributed UEs requiring a low-latency connection with an application in an NSI.
The NSE in its different variants can be optimized towards cost and revenue, latency or robustness by using the corresponding objective functions provided in this thesis. The objective functions optimize the overall utility of the embedded NSIs in first priority and propose an embedding with maximum revenue minus cost, minimum latency or maximum robustness in the second priority.

Beyond that, an efficient model and algorithm for NSE under resource uncertainty taking strict latency and real-time communication constraints into account are provided. The resource allocation is sensitive to resource uncertainties and optimizes the solution towards maximum resource confidences. It helps finding an NSE with the best balance between resource efficiency and robustness towards resource and demand fluctuations. The resource confidences of the calculated NSE solution are determined and can be assessed with regard to business policies.

A dedicated Java implementation of the developed NSE algorithms has been developed for the evaluation. The evaluation results show, that the proposed approach is a reliable mechanism for NSIA which can be applied to NSaaS. The presented solution is able to determine the feasibility of a new NSI-R received via the 5G Network Slice Customer Portal. Beyond that, it proposes a feasible, nearly optimal embedding of a single or a set of NSI-Rs. Furthermore, the resource and NSI confidences can be determined automatically by the software.
Small and medium sized problem instance can be solved within seconds or minutes on a regular notebook. In order to provide the best possible runtime performance, a linear objective function and linear constraints are used. In addition, the variables are boolean, except for link-to-path mapping variables in the path-splitting variant. Nevertheless, the complexity and scalability of the NSE problem remains challenging for large problem instances. The number of variables and constraints grows exponentially with the number of elements (nodes and links) in the physical network and the NSIs.

8.2 Future Work

The NSIA process as well as the presented models and algorithms for the NSE problem are based on accurate knowledge of the resources and capabilities provided by the mobile network and on the resource and capability requirements of the NSIs. Accurate and practically applicable methods for creating probabilistic models for the resource availability as well as the resource demand predictions for the operational as well as new NSIs are needed. However, this is out of scope of this thesis and should be addressed in future work.
This thesis focuses on modeling and solving the NSE problem. When advancing into practical applications of the NSIA, additional resources and capabilities might

be needed. Moreover, the size of the underlying network infrastructure and the number of NSIs might increase. The number of variables in the LP model grows over-proportionally, when the number of elements in the substrate and NSIs increase. Beyond that, the additional resources and capabilities lead to further constraints being added to the model. The preparation time of the model as well as the model solution time does not scale for large problem instances and the nearly optimal NSE algorithm cannot be completed in reasonable time for large problem instances, even on a high-performance computer. Thus, in future work, heuristics and preprocessing for making the NSE problem scalable for large problem instances and solving smaller ones even more efficiently should be evaluated.
One potential approach is splitting large problem instances into smaller subproblems. Either the problem can be split into local subproblem, or clusters of subnets can be abstracted and their resources can be aggregated. The first alternative can be implemented by splitting the substrate and the NSIs into subproblems corresponding to geographical areas. Solving all subproblems can be done in a fraction of the time needed for solving the full NSE problem. However, some central substrate nodes have to be shared among the subproblems. The second alternative requires a logical clustering and aggregation of resources, for instance, into subnets. The NSE problem can then be solved on a higher level of abstraction in the first solution step. In the second solution step, the details of the embedding within the clusters are determined. However, the resulting overall embedding might not be optimal due to the reduction of the assignment of the resource to the cluster or subnets in the first solution step.
Another potential approach to improve the runtime efficiency for large problem instances is to embed the NSIs in small groups or one after another. This highly reduces the number of constraints and embedding variables. However, this method requires to define an order in which the NSIs are embedded. The NSIs which are tried to be embedded first have a higher chance of being accepted. The NSI weights might be used to determine a good order for the NSIs. Nevertheless, it cannot be guaranteed that a nearly optimal solution is found.
Beyond that, promising heuristic approaches, that can be adjusted for the NSE problem are presented in literature. For instance, node-ranking based approaches, like [88] and the ViNE-Algorithm [89]. The accuracy, e.g., the achieved acceptance rates, of the heuristic solutions should be evaluated and compared to the accuracy of the nearly optimal algorithms presented in this thesis.

Another aspect of NSaaS not discussed in this works is that NSIs are often dynamic, i.e., they are deployed, run and terminated dynamically. Apart from that, NSIs can be activated during repeating time periods, for instance, every day during a specified time-window. Beyond that, the underlying physical network infrastructure is subject to persistent modification and change. This gives the resource allocation and planning performed with the proposed NSE algorithms an additional time dimension. The approach presented in this thesis can be enhanced to handle such dynamic NSE problems. A straightforward way to achieve this is to solve the NSE problem as described in this thesis for every time frame, i.e., after each change in the substrate network or in the set of NSIs. Certainly, this leads to a computational overhead for the NSE algorithm. A dynamic online reconfiguration of the resource allocation without the need to perform a full recalculation of the NSE algorithm is desirable.

Lastly, an even deeper analysis of resource overbooking should be done in order to improve the optimization towards the best balance between resource efficiency and robustness. One possibility for this is allowing fuzziness in the resource constraint fulfillment. In addition, customized probability distributions can be used for the different resources, for instance, gamma distributions, bounded normal distributions or customized discrete distributions.

Bibliography

[1] Cisco and affiliates. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022.* White Paper. Cisco and affiliates, Feb. 2019.

[2] Nokia Networks. *5G Masterplan – Five keys to create the new communications era.* White Paper. Nokia Networks, 2016.

[3] 3GPP. *Study on management and orchestration of network slicing for next generation network.* TR 28.801 V15.1.0. 3GPP, 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects. Jan. 2018.

[4] Matias Richart et al. "Resource Slicing in Virtual Wireless Networks: A Survey". In: *IEEE Transactions on Network and Service Management* 13.3 (Sept. 2016), pp. 462–476.

[5] 3GPP. *Management and orchestration of 5G networks: Concepts, use cases and requirements.* TS 28.530 V16.0.0. 3GPP, 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects. Sept. 2019.

[6] 3GPP. *Management and orchestration of 5G networks; Provisioning.* TS 28.531 V16.3.0. 3GPP, 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects. Sept. 2019.

[7] ETSI. *GS ZSM 002, Zero-touch network and Service Management (ZSM); Reference Architecture.* V1.1.1. Aug. 2019.

[8] Andrea Fendt et al. "A Network Slice Resource Allocation Process in 5G Mobile Networks". In: *Innovative Mobile and Internet Services in Ubiquitous Computing.* July 2018.

[9] Andrea Fendt et al. "A Network Slice Resource Allocation and Optimization Model for End-to-End Mobile Networks". In: *2018 IEEE 5G World Forum (5GWF).* July 2018, pp. 262–267.

[10] Andrea Fendt et al. "A Formal Optimization Model for 5G Mobile Network Slice Resource Allocation". In: *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).* Nov. 2018, pp. 101–106.

[11] Andrea Fendt et al. *An Apparatus and Method for Network Slice Instance Feasibility Checking in Network Slice Instantiation.* Nokia Bell Labs. Apr. 2019.

[12] Andrea Fendt et al. "An Efficient Model for Mobile Network Slice Embedding under Resource Uncertainty". In: *2019 16th International Symposium on Wireless Communication Systems (ISWCS).* 2019, pp. 602–606.

[13] Katha Ludwig, Andrea Fendt, and Bernhard Bauer. “An Efficient Online Heuristic for Mobile Network Slice Embedding”. In: *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. 2020, pp. 139–143.

[14] Andrea Fendt et al. “End-to-End Mobile Network Slice Embedding Leveraging Edge Computing”. In: *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. 2020, pp. 1–7.

[15] ETSI ZSM Nurit Sprecher. “ETSI ZSM Architectural Framework for End-to-End Service and Network Automation”. In: *IEEE.org, Software Defined Networks* (Nov. 2018).

[16] NGMN - Next Generation Mobile Networks Alliance. *5G End-to-End Architecture Framework*. Version 2.0.0. NGMN Alliance. Feb. 2018.

[17] ETSI. *GS ZSM 003, Zero-touch Network and Service Management (ZSM); End to end management and orchestration of network slicing*. V0.11.1. Oct. 2019.

[18] 3GPP. *Management and orchestration; Architecture framework*. TS 28.533 V16.1.0. 3GPP, 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects. Sept. 2019.

[19] NGMN - Next Generation Mobile Networks Alliance. *5G White Paper*. Version 1.0. NGMN Alliance. Feb. 2015.

[20] NGMN - Next Generation Mobile Networks Alliance. *Description of Network Slicing Concept*. Version 1.0.8 (draft). NGMN 5G P1 Requirements & Architecture Work Stream End-to-End Architecture. Sept. 2016.

[21] ETSI. *GS NFV 003, Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV*. V1.1.1. Oct. 2013.

[22] 3GPP. *System Architecture for the 5G System (5GS), Stage 2*. TS 23.501 V16.2.0. 3GPP, 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects. Sept. 2019.

[23] 5G Novel Radio Multiservice adaptive network Architecture. *Deliverable D3.2, 5G NORMA network architecture - Intermediate report*. Version 1.0. 5G NORMA. Jan. 2017.

[24] 5G Novel Radio Multiservice adaptive network Architecture. *Deliverable D3.3, 5G NORMA network architecture - Final report*. Version 1.0. 5G NORMA. Oct. 2017.

[25] GSM Association. *Generic Network Slice Template*. Version 2.0. Oct. 2019.

[26] GSM Association. *From Vertical Industry Requirements to Network Slice Characteristics*. Aug. 2018.

[27] Andreas Fischer et al. “Virtual Network Embedding: A Survey”. In: *IEEE Communications Surveys & Tutorials* 15.4 (2013), pp. 1888–1906.

[28] David G. Andersen. *Theoretical Approaches to Node Assignment*. unpublished. Dec. 2002.

[29] S. G. Kolliopoulos and C. Stein. “Improved approximation algorithms for unsplittable flow problems”. In: *Proceedings 38th Annual Symposium on Foundations of Computer Science*. Miami Beach, FL, USA: IEEE Comput. Soc, 1997, pp. 426–436.

[30] Matthias Rost and Stefan Schmid. “NP-Completeness and Inapproximability of the Virtual Network Embedding Problem and Its Variants”. In: *Cornell University* arXiv.og, 1801.03162 (Mar. 28, 2018).

[31] Zoran Despotovic et al. “VNetMapper: A fast and scalable approach to virtual networks embedding”. In: *Proceedings - International Conference on Computer Communications and Networks, ICCCN*. Aug. 2014, pp. 1–6.

[32] Andreas Fischer et al. “ALEVIN - A Framework to Develop, Compare, and Analyze Virtual Network Embedding Algorithms”. In: *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.* 37 (2011), p. 13.

[33] David Stezenbach, Matthias Hartmann, and Kurt Tutschku. “Parameters and challenges for Virtual Network embedding in the Future Internet”. In: *2012 IEEE Network Operations and Management Symposium*. 2012 IEEE/IFIP Network Operations and Management Symposium (NOMS 2012). Maui, HI: IEEE, Apr. 2012, pp. 1272–1278.

[34] Zhongbao Zhang et al. “A unified enhanced particle swarm optimization-based virtual network embedding algorithm”. In: *International Journal of Communication Systems* 26.8 (Aug. 2013), pp. 1054–1073.

[35] Kailing Guo et al. “Particle swarm optimization based multi-domain virtual network embedding”. In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. Ottawa, ON, Canada: IEEE, May 2015, pp. 798–801.

[36] Ashraf A. “Memetic Multi-Objective Particle Swarm Optimization-Based Energy-Aware Virtual Network Embedding”. In: *International Journal of Advanced Computer Science and Applications* 6.4 (2015).

[37] Ilhem Fajjari et al. “VNE-AC: Virtual Network Embedding Algorithm Based on Ant Colony Metaheuristic”. In: *2011 IEEE International Conference on Communications (ICC)*. Kyoto: IEEE, June 2011, pp. 1–6.

[38] Haotong Cao et al. “Exact solutions of VNE: A survey”. In: *China Communications* 13.6 (June 2016), pp. 48–62.

[39] Haotong Cao et al. “Heuristic solutions of virtual network embedding: A survey”. In: *China Communications* 15.3 (Mar. 2018), pp. 186–219.

[40] Laurence A. Wolsey. *Integer Programming*. Wiley & Sons, 1998.

[41] CPLEX Optimizer. URL: `https://www.ibm.com/uk-en/analytics/cplex-optimizer` (visited on 01/15/2021).

[42] *FICO Xpress Optimization*. 2019. URL: `https://www.fico.com/de/products/fico-xpress-optimization` (visited on 01/15/2021).

[43] *Gurobi Optimization.* URL: `https://www.gurobi.com/products/gurobi-optimizer/` (visited on 01/15/2021).

[44] *GLPK (GNU Linear Programming Kit).* 2012. URL: `https://www.gnu.org/software/glpk/glpk.html` (visited on 01/15/2021).

[45] Sebastian Pokutta et al. *SCIP Solving Constraint Integer Programs.* 2019. URL: `https://scip.zib.de` (visited on 01/15/2021).

[46] *Parallel and Distributed Optimization with Gurobi.* URL: `https://www.gurobi.com/resource/parallel-and-distributed-optimization/` (visited on 01/15/2021).

[47] *IBM Knowledge Center, Remote object for distributed parallel optimization.* URL: `https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/parallel_optim/remote_obj/00_remote_obj_synopsis.html` (visited on 01/15/2021).

[48] GNU. *GNU Linear Programming Kit, Reference Manuual.* Version for GLPK Version 4.64. Nov. 2017.

[49] Ambros Gleixner et al. *The SCIP Optimization Suite 6.0.* Technical Report. Optimization Online, July 2018. URL: `http://www.optimization-online.org/DB_HTML/2018/07/6692.html`.

[50] Matias Richart et al. "Resource Slicing in Virtual Wireless Networks: A Survey". In: *IEEE Transactions on Network and Service Management* 13.3 (Sept. 2016), pp. 462–476.

[51] Reinhard Diestel. *Graphentheorie.* 3., neu bearb. und erw. Aufl. Berlin: Springer, 2006.

[52] Sami Kekki et al. "MEC in 5G networks". In: *ETSI White Paper No. 28* (June 2018).

[53] Sebastian Pokutta et al. *SCIP Solving Constraint Integer Programs.* 2020. URL: `%7Bhttps://scipopt.org/doc/html/PARAMETERS.php%7D` (visited on 01/15/2021).

[54] Roberto Riggio et al. "Scheduling Wireless Virtual Networks Functions". In: *IEEE Transactions on Network and Service Management* 13.2 (June 2016), pp. 240–252.

[55] Ilias Tsompanidis, Ahmed H. Zahran, and Cormac J. Sreenan. "A Utility-Based Resource and Network Assignment Framework for Heterogeneous Mobile Networks". In: *2015 IEEE Global Communications Conference (GLOBECOM).* 2015, pp. 1–6.

[56] Flavio Esposito, Donato Di Paola, and Ibrahim Matta. "A general distributed approach to slice embedding with guarantees". In: *2013 IFIP Networking Conference, IFIP Networking 2013.* Jan. 2013, pp. 1–9.

[57] Mao Yang et al. "Karnaugh-map like online embedding algorithm of wireless virtualization". In: *The 15th International Symposium on Wireless Personal Multimedia Communications.* 2012, pp. 594–598.

[58] Haijun Zhang et al. "Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges". In: *IEEE Communications Magazine* 55.8 (2017), pp. 138–145.

[59] Gang Wang et al. "Resource Allocation for Network Slices in 5G with Network Resource Pricing". In: *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. 2017, pp. 1–6.

[60] Menglan Jiang, Massimo Condoluci, and Toktam Mahmoodi. "Network slicing management prioritization in 5G mobile systems". In: *European Wireless 2016; 22th European Wireless Conference*. 2016, pp. 1–6.

[61] Spyridon Vassilaras et al. "The Algorithmic Aspects of Network Slicing". In: *IEEE Communications Magazine* 55.8 (Aug. 2017), pp. 112–119.

[62] Tachporn Sanguanpuak et al. "Network Slicing with Mobile Edge Computing for Micro-Operator Networks in Beyond 5G". In: *2018 21st International Symposium on Wireless Personal Multimedia Communications (WPMC)*. Chiang Rai, Thailand: IEEE, Nov. 2018, pp. 352–357.

[63] Bin Xiang et al. "Joint Network Slicing and Mobile Edge Computing in 5G Networks". In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. Shanghai, China: IEEE, May 2019, pp. 1–7.

[64] Sooeun Song and Jong-Moon Chung. "Sliced NFV service chaining in mobile edge clouds". In: *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. Sept. 2017, pp. 292–294.

[65] Mansoor Alicherry and T.V. Lakshman. "Network aware resource allocation in distributed clouds". In: *2012 Proceedings IEEE INFOCOM*. IEEE INFOCOM 2012 - IEEE Conference on Computer Communications. Mar. 2012, pp. 963–971.

[66] Fang Hao et al. "Online Allocation of Virtual Machines in a Distributed Cloud". In: *IEEE/ACM Transactions on Networking* 25 (July 2016), pp. 1–12.

[67] Nabeel Akhtar et al. "Virtual Function Placement and Traffic Steering over 5G Multi-Technology Networks". In: *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. Montreal, QC: IEEE, June 2018, pp. 114–122.

[68] Weisheng Xie et al. "Network virtualization with dynamic resource pooling and trading mechanism". In: *2014 IEEE Global Communications Conference*. Dec. 2014, pp. 1829–1835.

[69] Markus Fiedler. "On Resource Sharing and Careful Overbooking for Network Virtualization". In: *20th ITC Specialist Seminar* (May 2009).

[70] Faruk Çağlar and Aniruddha Gokhale. "iOverbook: Intelligent Resource-Overbooking to Support Soft Real-Time Applications in the Cloud". In: *2014 IEEE 7th International Conference on Cloud Computing*. 2014 IEEE 7th International Conference on Cloud Computing. June 2014, pp. 538–545.

[71] Luis Tomás and Johan Tordsson. “Cloud Service Differentiation in Overbooked Data Centers”. In: *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. Dec. 2014, pp. 541–546.

[72] Tianyu Wo et al. “Overbooking-Based Resource Allocation in Virtualized Data Center”. In: *2012 IEEE 15th International Symposium on Object/Component/ Service-Oriented Real-Time Distributed Computing Workshops*. Apr. 2012, pp. 142–149.

[73] David Hoeflin and Paul Reeser. “Quantifying the performance impact of overbooking virtualized resources”. In: *2012 IEEE International Conference on Communications (ICC)*. June 2012, pp. 5523–5527.

[74] Jungmin Son et al. “SLA-Aware and Energy-Efficient Dynamic Overbooking in SDN-Based Cloud Data Centers”. In: *IEEE Transactions on Sustainable Computing* 2.2 (Apr. 2017), pp. 76–89.

[75] Robert Ball et al. “Aggressive telecommunications overbooking ratios”. In: *IEEE International Conference on Performance, Computing, and Communications, 2004*. Apr. 2004, pp. 31–38.

[76] Jianming Liu, Xiaohong Jiang, and Susumu Horiguchi. “Opportunistic link overbooking for resource efficiency under per-flow service guarantee”. In: *IEEE Transactions on Communications* 58.6 (June 2010), pp. 1769–1781.

[77] Zhaleh Sadreddini, Erkan Guler, and Tuğrul Çavdar. “PSO-optimized Instant Overbooking Framework for cognitive radio networks”. In: *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*. July 2015, pp. 49–53.

[78] Antonio Marotta and Andreas Kassler. “A Power Efficient and Robust Virtual Network Functions Placement Problem”. In: *2016 28th International Teletraffic Congress (ITC 28)*. Würzburg, Germany: IEEE, Sept. 2016, pp. 331–339.

[79] Antonio Marotta et al. “A fast robust optimization-based heuristic for the deployment of green virtual network functions”. In: *Journal of Network and Computer Applications* 95 (Oct. 2017), pp. 42–53.

[80] Bego Blanco et al. “A Robust Optimization Based Energy-Aware Virtual Network Function Placement Proposal for Small Cell 5G Networks with Mobile Edge Computing Capabilities”. In: *Mobile Information Systems* 2017 (2017), pp. 1–14.

[81] Ayşegül Altın et al. “Provisioning virtual private networks under traffic uncertainty”. In: *Networks* 49 (Jan. 2007), pp. 100–115.

[82] Giorgos Chochlidakis and Vasilis Friderikos. “Robust virtual network embedding for mobile networks”. In: *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. Sept. 2015, pp. 1867–1871.

[83] Varun Reddy, Andreas Baumgartner, and Thomas Bauschert. “Robust embedding of VNF/service chains with delay bounds”. In: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. Nov. 2016, pp. 93–99.

[84] Andreas Baumgartner et al. “Network slice embedding under traffic uncertainties - A light robust approach”. In: *2017 13th International Conference on Network and Service Management (CNSM)*. Tokyo: IEEE, Nov. 2017, pp. 1–5.

[85] Stefano Coniglio, Arie Koster, and Martin Tieves. “Virtual network embedding under uncertainty: Exact and heuristic approaches”. In: *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*. Mar. 2015, pp. 1–8.

[86] Stefano Coniglio, Arie Koster, and Martin Tieves. “Data Uncertainty in Virtual Network Embedding: Robust Optimization and Protection Levels”. In: *Journal of Network and Systems Management* 24 (July 2016), pp. 681–710.

[87] Tri Trinh, Hiroshi Esaki, and Chaodit Aswakul. “Quality of service using careful overbooking for optimal virtual network resource allocation”. In: *The 8th Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology (ECTI) Association of Thailand - Conference 2011*. 2011, pp. 296–299.

[88] Xiang Cheng et al. “Virtual network embedding through topology-aware node ranking”. In: *ACM SIGCOMM Computer Communication Review* 41.2 (Apr. 15, 2011), p. 38.

[89] Mosharaf Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. “ViNE-Yard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping”. In: *IEEE/ACM Transactions on Networking* 20.1 (2012), pp. 206–219.

List of Acronyms

3GPP	3rd Generation Partnership Project
AI	Artificial Intelligence
AMF	Access and Mobility Management Function
API	Application Programming Interface
BSS	Business Support System
BTS	Base Transceiver Station
CAGR	Compound Annual Growth Rate
CP	Control Plane
CPU	Central Processing Unit
CQI	Channel Quality Index
CU-CP	Central Unit - Control Plane
CU-UP	Central Unit - User Plane
DU	Distributed Unit
E2E	End-to-End
eMBB	enhanced Mobile Broadband
ETSI	European Standards Organization
EVR	Evaluation Result
FDMA	Frequency Division Multiple Access
GLPK	GNU Linear Programming Kit
gNB	Next Generation Node Base Station
GSMA	Groupe Speciale Mobile Association
GST	Generic Network Slice Template
ILP	Integer Linear Program
ISG	Industry Specification Group
InP	Infrastructure Provider

IoT	Internet of Things
KPI	Key Performance Indicator
LP	Linear Program
LTE	Long Term Evolution
M2M	Machine-to-Machine
MANO	Management and Orchestration
MBB	Mobile Broadband
MD	Management Domain
MILP	Mixed Integer Linear Proram
MIP	Mixed Integer Program
MAI	Multiple Application Instantiation
MEC	Mobile Edge Computing
MAEC	Multi-Access Edge Computing
MIMO	Multiple Input Multiple Output
ML	Machine Learning
MSP	Mobile Service Provider
MNO	Mobile Network Operator
mMTC	massive Machine Type Communication
MTC	Machine Type Communication
NEST	Network Slice Type
NF	Network Function
NFV	Network Function Virtualization
NGMN	Next Generation Mobile Networks
NSaaS	Network Slice as a Service
NSE	Network Slice Embedding
NSI	Network Slice Instance
NSIA	Network Slice Instance Admission

NSI-D	Network Slice Instance Description
NSI-FC	Network Slice Instance Feasibility Checker
NSI-P	Network Slice Instance Provider
NSI-R	Network Slice Instance Request
NSMS	Network Slice Management Service
NSP	Network Slice Provider
NSS	Network Slice Subnet
NSSI	Network Slice Subnet Instance
NSSMS	Network Slice Subnet Management Service
OPEX	Operating Expenditures
OTT	Over-The-Top
RAT	Radio Access Technology
PCF	Policy Control Function
PDU	Packet Data Unit
PNF	Physical Network Function
POF	Probability of Feasibility
PRB	Physical Resource Block
QoE	Quality of Experience
QoS	Quality of Service
QP	Quadratic Program
RAM	Random Access Memory
RAN	Radio Access Network
SCIP	Solving Constraint Integer Programs
SDN	Software Defined Network
SLA	Service Level Agreement
SMF	Session Management Function
SNIR	Signal-to-Noise-plus-Interference Ratio

STD	Standard Deviation
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
TN	Transport Network
UE	User Equipment
UP	User Plane
UPF	User Plane Function
URLLC	Ultra-Reliable and Low Latency Communication
V2V	vehicle-to-vehicle
V2X	vehicle-to-everything
VNE	Virtual Network Embedding
VM	Virtual Machine
VNF	Virtualized Network Function
ZSM	Zero touch network and Service Management

List of Symbols

G	Undirected graph
$\mathcal{V}$	Set of vertices/nodes
v	Vertex/node
$\mathcal{E}$	Set of links/edges
e	Link/edge
P	Path in a graph
$\mathcal{P}$	Set of communication paths
$\mathcal{P}_{v_i,v_j}$	Set of paths connecting two vertices v_i, v_j
N	Substrate network graph
N_k	k-th network slice
$\mathcal{U}$	Set of UE groups
u	UE group
$\mathcal{C}$	Set of cloud/server/router nodes
c	Cloud/server/router node
$\mathcal{A}$	Set of applications/NFs/services
a	Application/NF/service node/vertex in network slice
$\mathcal{L}$	Set of links/edges in network slice
l	Link/edge in network slice
O_w	Provided node resource on w-th node in the substrate
D_w	Provided computation capacity on w-th node in the substrate

M_w	Provided memory capacity on w-th node in the substrate
W_w	Provided capability on the w-th node in the substrate
B_w	Provided reliability on the w-the node in the substrate
O_m^k	Required node resource on m-th node in the k-th network slice
D_m^k	Required computation capacity of m-th node in k-th network slice
M_m^k	Required memory capacity of m-th node in k-th network slice
W_m^k	Required capability of the m-th node in the k-th network slice
B_m^k	Required reliability of the m-th node in the k-th network slice
R_j	Provided link resource on j-th link in the substrate
T_j	Provided throughput capacity on j-th link in the substrate
Q_j	Provided capability of the j-the edge in the substrate
L_j	Maximum latency on j-the edge in the substrate
A_j	Availability of j-th edge in the substrate
R_i^k	Required link resource on i-th link of the k-th network slice
T_i^k	Required throughput of i-th link of the k-th network slice
Q_i^k	Required capability of the i-the link of the k-th network slice
L_i^k	Allowed latency on i-th link of the k-the network slice
A_i^k	Required reliability of i-th link of the k-th network slice
R_{P_r}	Provided resource of substrate path P_r
T_{P_r}	Provided throughput of substrate path P_r
Q_{P_r}	Provided capability of substrate path P_r
L_{P_r}	Latency on substrate path P_r
A_{P_r}	Reliability of substrate path P_r

$\mu_{O_m^k}$	Mean of required resource of a_m^k in N_k
$\sigma_{O_m^k}$	STD of required resource of a_m^k in N_k
$\mu_{D_m^k}$	Mean of required computation capacity of a_m^k in N_k
$\sigma_{D_m^k}$	STD of required computation capacity of a_m^k in N_k
$\mu_{M_m^k}$	Mean required memory capacity of a_m^k in N_k
$\sigma_{M_m^k}$	STD of required memory capacity of a_m^k in N_k
μ_{R_j}	Mean of provided resource of substrate edge e_j
σ_{R_j}	STD of provided resource of substrate edge e_j
μ_{T_j}	Mean of provided throughput of substrate edge e_j
σ_{T_j}	Mean, STD of provided throughput of substrate edge e_j
$\mu_{R_i^k}$	Mean of required resource of link l_i^k in N_k
$\sigma_{R_i^k}$	STD of required resource of link l_i^k in N_k
$\mu_{T_i^k}$	STD of required throughput of link l_i^k in N_k
$\sigma_{T_i^k}$	Mean, STD of required throughput of link l_i^k in N_k
POF_{O_w}	POF for general node resource O_w
POF_{D_w}	POF for CPU node resource D_w
POF_{M_w}	POF for memory node resource M_w
POF_{R_j}	POF for general link resource R_j
POF_{T_j}	POF for throughput link resource T_j
POF_D^k	Combined POF for all CPU node resources of N_k
POF_M^k	Combined POF for all memory node resources of N_k
POF_T^k	Combined POF for all throughput link resources of N_k
POF^k	Combined POF for all CPU node resources of N_k

y_k	Binary variables of embedding k-th network slice into the substrate
$a2c^k_{m,w}$	Binary variables of embedding m-th application node of k-th network slice onto the w-th cloud node in the substrate
$l2p^k_{i,r}$	Variables of embedding the i-th link of the k-th network slice on the r-th path in the substrate
$p2e_{r,j}$	Binary parameters of the association between the r-th communication path in the substrate and the j-th edge in the substrate
$l2e^k_{i,j}$	Derived binary variable of containment of the j-th edge in the substrate in a path, which the i-th link of the k-th network slice is mapped to
ω	Network slice weight/relative importance
ϕ	NSI revenue normalization factor
ψ	NSI cost normalization factor
ρ	Weighting factor in ILP objective functions
γ	Safety buffer in NSE under uncertainty

List of Definitions

List of Figures

List of Tables

List of Models

List of Algorithms