PhD-FSTM-2023-104

The Faculty of Science, Technology and Medicine

DISSERTATION

Presented on 05/10/2023 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITE DU LUXEMBOURG
EN INFORMATIQUE

by

Mario MINARDI
Born on 11 January 1996 in Bologna, Italy

TRAFFIC ENGINEERING ALGORITHMS FOR SOFTWARE DEFINED SATELLITE-TERRESTRIAL NETWORKS

Dissertation defense committee

Dr. Symeon Chatzinotas, Dissertation Supervisor
Professor and Head of SIGCOM, SnT, University of Luxembourg
Dr. Fabrizio Pastore, Chairman
Assistant Professor, SnT, University of Luxembourg
Dr. Giovanni Gianbene, Vice-Chairman
Assistant Professor, University of Siena
Dr. Christos Politis,
Senior Engineer, Communication Systems, SES Techcom S.A., Luxembourg
Dr. Ramon Ferrus,
Professor, UTC, Spain

Traffic Engineering Algorithms for Software Defined Satellite-Terrestrial Networks

Mario Minardi

Abstract

The telecom industry foresaw a constant change over the decades. Alongside the standardization of 5G, not only the traffic demand increased, but also the different requirements of the provided services, e.g., latency, traffic load and pattern, datarate. To match this trend, the traditional telecom infrastructure has been revolutionized, going from the "one-size-fits-all" model to a shared approach, where the infrastructure is shared among enterprise customers with even very different requirements. In this context, a paradigm, named network slicing, has considerably attracted the network operators. Network slicing is a successful enabling model for the 5G and beyond networks because it allows operators to tailor their networks based on the end use case, release unused functionalities and resources and dynamically assign them to different customers.

In addition, 6G networks are advertised to bring the terrestrial and satellite networks closer, to work in a coordinated way and provide ubiquitous, heterogenous and reliable services. In this context, this thesis investigates the optimization of network slicing in an integrated satellite-terrestrial network. Well-known enabling technologies for network slicing, such as Software-Defined Networking (SDN), are included as a proof-of-concept SDN-based testbed to demonstrate and support the proposed optimization algorithms. As network slicing is a resource allocation problem, where virtualized resources are accommodated on the substrate network, we investigate this optimization problem that is well-known in the literature as Virtual Network Embedding (VNE).

Firstly, we study the application of VNE to an integrated Medium Earth Orbit (MEO)-terrestrial network with the objective of minimizing the traffic migrations, considering the existence of inter-satellite links (ISLs) too. As satellite handovers are unavoidable, we showed as including the minimization of traffic handovers in the objective function, brings to a gain in terms of traffic migrations and packet loss up to 2.5-5% compared to the traditional approaches.

Secondly, we investigated the benefit of flexibly accommodating traffic demands without fully assigning the required resources while keeping the user satisfaction probability (USP) under control. Thanks to the SDN-based testbed, the traffic is generated and the statistics are collected to real-time match the need of each user. This showed an increase in the acceptance ratio up to 11% compared to the baselines.

Lastly, as the previous two main chapters investigated the point-to-point connectivity, we expanded the work to the embedding of full slices for a combined Geostationary (GEO)-Low Earth Orbit (LEO) satellites and terrestrial network. We provided a flexible framework, for 6G use-cases with real network requirements, which operates based on prioritization, minimizes the migrations of slices when congestions occur over the substrate network and proactively manages the satellite handovers for each slice.

Finally, we discuss conclusive remarks and future research directions.

Acknowledgements

First of all, I would like to deeply thank Prof. Symeon Chatzinotas for being the mentor of this thesis and continuously providing useful and meaningful suggestions, comments and feedbacks which have helped the realization of this thesis. Together with him, I'm thankful to Dr. Thang X. Vu, Dr. Ilora Maity and Dr. Youssouf Drif for daily following the progress and helping/guiding me through this journey.

An additional thank goes to the CET members, Dr. Christos Politis and Prof. Ramon Ferrus for providing their external feedbacks during the CET which have also increased the quality of the works presented in this thesis.

This experience has also allowed me to meet wonderful friends and colleagues, and among them, thanks to the ones I have shared most of my time, starting from who I have started this journey with, Clément and Raphaël, and later on the way, Carla and my cool kids. A special thanks goes to Eugene and Giacomo who had to daily deal with my desperations/successes/failures during the last year(s). Last, but not least, my most gratitude is for Marta. From almost the beginning of my journey till the very end, she shared with me all the feelings of this incredible rollercoster experience and it would have been much more difficult without her daily support and comprehension.

Mario Minardi

Luxembourg, 5^{th} October 2023

Preface

This Ph.D. thesis has been carried out from March 2020 to October 2023, at the SIGCOM research group, SnT - Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, under the supervision of Prof. Symeon Chatzinotas, Dr. Thang X. Vu and Dr. Christos Politis (SES Techcom). The time-to-time evaluation of the Ph.D. thesis was duly performed by the CET members constituting the supervisors and co-supervisors.

Contents

This Ph.D. thesis entitled "Traffic Engineering Algorithms for Software Defined Satellite-Terrestrial Networks" is divided into six chapters. In Chapter 1, the introduction, background, motivation, and contributions of this thesis are described. Chapter 2 provides the description of the SDN-based testbed developed and used during the PhD. Chapters 3, 4 and 5 provide the three main contributions of this thesis. Finally, Chapter 6 provides concluding remarks and future work.

Collaborations

This work was made possible thanks to the collaboration with SES Techcom S.A.

Support of the Thesis

This work was supported by the Luxembourg National Research Fund (FNR) in the IS - Information and Communication Technologies domain under Industrial Partnership Program with industrial partner SES S.A., Industrial Partnership Block Grant (IPBG), ref. FNR/IPBG19/14016225/INSTRUCT project title "ASWELL- AutonomouS NetWork Slicing

for IntEgrated SateLlite-TerrestriaL Transport Networks," grant FNR/C19/ IS/13718904/ ASWELL/Chatzinotas.

Declaration

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of the authors knowledge, original and has not been submitted in whole or part for a degree in any university.

Publication List

Journals

[I] M. Minardi, T. X. Vu, L. Lei, C. Politis and S. Chatzinotas, "Virtual Network Embedding for NGSO Systems: Algorithmic Solution and SDN-Testbed Validation," *in IEEE Transactions on Network and Service Management,* 2022, doi: 10.1109/TNSM.2022.3225748.

[II] M. Minardi, T. X. Vu, L. Lei, I. Maity, C. Politis, S. Chatzinotas, "Traffic-Aware Virtual Network Embedding with Joint Load Balancing and Resource Assignment", *IEEE Transactions on Networks and Service Management* - under major revision.

[III] M. Minardi, Y. Drif, T. X. Vu and S. Chatzinotas, "SAST-VNE: A Flexible Framework for Network Slicing in Beyond-5G Integrated Satellite-Terrestrial Networks," - submitted to *IEEE Journal on Selected Areas in Communications (JSAC).*

Conference/Workshop Papers

[IV] M. Minardi, S. K. Sharma, S. Chatzinotas and T. X. Vu, "A Parallel Link Mapping for Virtual Network Embedding with Joint Load-Balancing and Energy-Saving," 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), Athens, Greece, 2021, pp. 419-424, doi: 10.1109/MeditCom49071.2021.9647664.

[V] M. Minardi, Y. Drif, T. X. Vu, I. Maity, C. Politis and S. Chatzinotas, "SDN-based Testbed for Emerging Use Cases in Beyond 5G NTN-Terrestrial Networks," NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, Miami, FL, USA, 2023,

pp. 1-6, doi: 10.1109/NOMS56928.2023.10154319.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
CAPEX	Capital Expenditure
DTA-VNE	Dynamic topology-aware-VNE
eMBB	enhanced Mobile BroadBand
ESA	European Space Agency
E2E	End-to-end
GEO	Geostationary Earth Orbit
GSO	Geostationary orbit
GUI	Graphical user interface
GW	Gateway
HAP	High altitude platform
HEO	High elliptical orbit
IaaS	Infrastructure as a Service
InP	Infrastructure provider
IP	Internet Protocol
ISL	Inter-satellite links
ISO/OSI	International Organization for Standardization/Open Systems Interconnection
LEO	Low Earth Orbit
LoS	Line of sight
LP	Linear programming
MAC	Medium access control
MBLP	Mixed Binary Linear Programming
MEO	Medium Earth Orbit

MIRSAT	MultI-layer awaRe SDN-based testbed for SAtellite-Terrestrial networks
ML	Machine Learning
mMTC	massive machine type comunications
NFV	Network Function Virtualization
NGSO	Non Geostationary Orbit
NTN	Non-terrestrial network
OPEX	Operational Expenditur
QoS	Quality of Service
RAM	Random Access Memory
REST	Representational state transfer
RTT	Round trip time
SDN	Software-Defined Networking
SNR	Signal to Noise Ratio
SP	Service Provider
STK	System Tool Kit
TCP	Transmission control protocol
UDP	User datagram protocol
URLLC	ultra-reliable and low-latency communications
VLAN	Virtual local area network
VNE	Virtual Network Embedding
VNO	Virtual Network Operator
VNP	Virtual Network Provider
VNR	Virtual Network Request

Chapter 1

Introduction

The telecom sector of today is heavily impacted by the exponential increase in traffic demands, the number of users and the diversity of applications that has characterized the last decade (1). As estimated (2), by the end of 2023, the internet will be accessible by more than 5 billion users, i.e., around two thirds of the global population. The growing demand for more capacity has revolutionized the network infrastructure. While terrestrial and satellite networks were two separate realities, with different objectives, delivered services and targeted markets, with 5G the combination between the two was already proposed. The intent was to mitigate the need for the use cases via the combination of different QoS (Quality of Service), such as latency, capacity, on-board computing capability, infrastructure cost, ubiquitous coverage, that either of the two could provide. In addition to that, the trend of minimizing the cost and increasing the revenues of the infrastructure and service provider, drove the interest towards a virtualization process of the physical resources, which reduces both Capital Expenditure (CAPEX) and Operational Expenditure (OPEX). In this context, network slicing is one of the key techniques that provides a layered version of the physical network as a set of multiple logical networks to support a wide range of diverse applications. As this revolutionized approach requires time to be operative, beyond 5G networks are expected to further push terrestrial and satellite companies to closely work together with the common objective of delivering a seamless integrated network. In this context, it is clear that the complexity of both network and traffic management increases. The scope of this thesis is to investigate, on one side, the optimization of the shared network infrastructure applied to a dynamic substrate network, such as the integrated satellite-terrestrial one. On the other hand, this thesis proposes some practical implementations of well-established tools, such as software-

defined networking, to show the application of the proposed algorithms to real scenarios.

1.1 Background

This section intends to provide the basic concepts to understand the contribution of this thesis, starting from the business model of a telecom network, the role and benefit of the satellite integration with the terrestrial network, the well-known resource allocation algorithm, named Virtual Network Embedding (VNE), network slicing and the principles of software-defined networking.

1.1.1 The network business model

The Internet Service Providers (ISPs) were traditionally a single entity with the task of managing the infrastructure and providing end-to-end services to users. Already at the beginning of the previous decade (3; 4), the network virtualization was seen as the key to overcome the complexity of the internet. The advent of network virtualization has triggered a new business model that divides the traditional roles of ISPs into two main entities: Infrastructure Providers (InPs) and Service Providers (SPs). The InP is responsible for the acquisition and maintenance of the physical hardware and the software running on it. The SP, as customer of the InP, pays to use the infrastructure and the resources provided by the InP, manages the communication protocols and applications and provide the end-to-end connectivity. The virtualization process has modified the traditional structure of the Internet SP, as shown in Figure 1.1, with the novel concept of Infrastructure as a Service (IaaS). It is important to underline that the main and basic element of virtualized internet are defined as Virtual Network Requests (VNRs). VNRs are seen as subgraph, combination of interconnected nodes and links that is mapped to the physical, or substrate, network. While the infrastructure provider is responsible for the physical devices, their interconnection and maintenance, the SP is layered into three main players. The closest to the InP is the Virtual Network Provider (VNP), which is responsible for abstracting and assembling the physical resources provided by one or more InPs. Then, once the physical resources are ready to be assigned, the Virtual Network Operator (VNO) takes over with the task of managing the installation and operation of the VNRs, depending on the requirements of the SP. Finally, the SP is the customer-oriented player and is responsible for the business of the VNR as customized services.

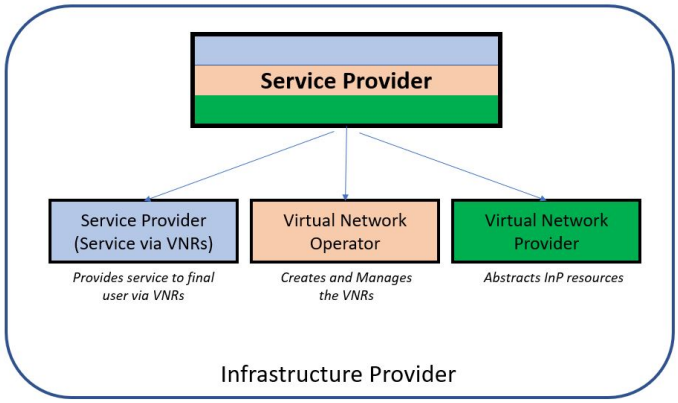


Figure 1.1: Traditional online VNE solution for dynamic substrate network

As the substrate network has limited resources, the embedding of VNRs (topology and resources) onto the substrate network is a well-known resource allocation problem, the VNE.

1.1.2 Network Slicing

5G networks are the first standardized transport networks that were originally designed to support and implement network slicing (5; 6; 7), even though the industry has followed a more cautious and shy approach to this revolution (8). The main idea comes from the need to use a single infrastructure for differentiated services, in terms of QoS and resources requirements, e.g. tolerated latency, computing capabilities and datarate. Network slicing is the concept of subdividing the infrastructure layer into sub-layers. As included in the standardization document (9) from 3GPP, a network slice is defined as "A logical network that provides specific network capabilities and network characteristics". Additionally, in the specification (10), the features of the slice are described such as "Priority, end-to-end latency, service availability requirement". From the network virtualization perspective, instead, a VNR, as mentioned earlier, is the combination of active and passive network elements (network nodes and network links) on top of a substrate network. From a mathematical perspective, this corresponds to subdivide the main graph (the original infrastructure), with the resources available for each link and node, into smaller subgraphs, and assign each one its own set of nodes, links and the respective resources. This can be done in a static way, where the sub-layers are pre-defined and services can only use the included resources. However, clearly, the flexible way is always the most convenient solution, as proposed in the current Beyond 5G (B5G) and 6G networks. This means, for each sub-layer, there is a pool of resources initially

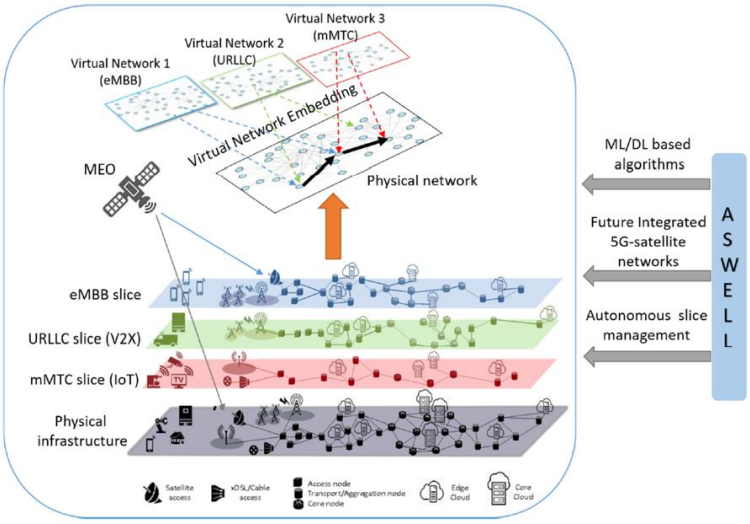


Figure 1.2: Network slicing

assigned, but, eventually, depending on the traffic conditions or demands, resources can be switched from one sub-layer to another one. Network slicing has gained particular interest already in 5G and even more in B5G, due to the explosion of the well-known novel use-cases and traffic differentiation, e.g., enhanced mobile broadband (eMBB), ultra-reliable and low latency communications (URLLC) and massive machine type comunications (mMTC). While the traditional approach foresaw separated infrastructures for different services, network slicing strongly relies on the flexible assignment of the infrastructure and to its adaptation to the traffic demands.

It is well understood that an optimized slicing allows to fully exploit, if the optimization does not fall into heuristic solutions which oversimply the solution, the infrastructure resources. For this reason, network slicing is strongly supported by optimization algorithms, known under the name of VNE, as mentioned in the previous section, that takes as input the slice, which we can refer to as VNR to keep consistency with the VNE terminology, and finds the optimal embedding depending on the objective of the optimization, e.g., load-balancing or shortest-path. Different national and international-funded projects investigated the network slicing application to novel transport networks such as Autonomous Network Slicing for Integrated Satellite-Terrestrial Transport Networks (ASWELL) (11), see Figure 1.2, SLICENET (12), CORDIS (13), the initiatives from the European 5G observatory (14) and the European 5G group (15). Some relevant key enabling technologies for network slicing have already been proposed in the literature such as software-defined networking (SDN) and network function

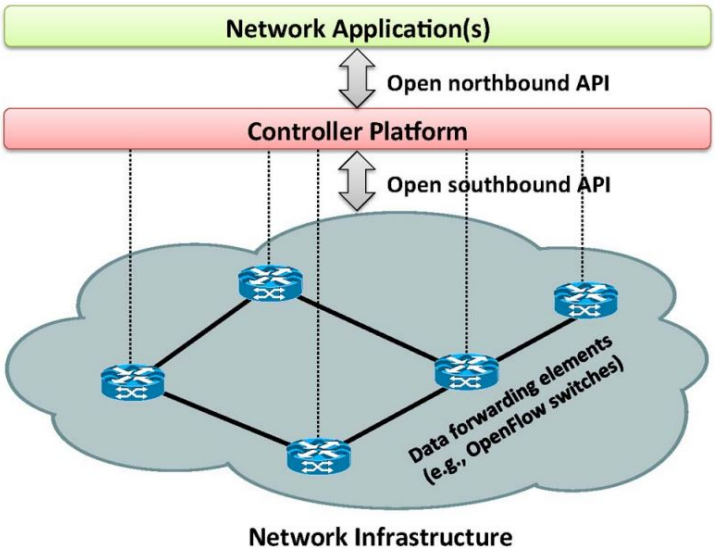


Figure 1.3: Software Defined Networking

virtualization (NFV). We introduce their details and discussion in the following section.

1.1.3 Software Defined Networking (SDN) and Network Function Virtualization (NFV)

The success of Software-Defined Networking (SDN) (16) dates back to the previous decade, when the traditional Internet Protocol (IP) networks began to show some weakness in adapting to the fast growth of the internet and transport networks in general. Traditional IP networks rely on distributed network protocols in switches and routers to deliver packets from source to destination through intermediate steps of the network. With the increase in dynamicity and size of transport networks, this distributed approach has shown weakness and a slow pace to adapt to that. In addition, switches and routers are physical devices which typically are a vendor-based solution. As the programmability of each device depends on its interfaces and programming language, this considerably slowed down the process of implementing new devices into an already built network. The last, but not least, weakness is related to the real-time situations, such as congestion and failures, as traditional IP networks lack automatic reconfiguration mechanisms to promptly recover from that.

To address the weaknesses mentioned above, a centralized approach, known as SDN, was proposed. In fact, SDN is based on decoupling the data plane from the control plane. The control plane is constituted by the SDN controller which has full view and knowledge of the

substrate network. One of the greatest novelty and advantages of SDN is that the network can simply become a set of interconnected programmable switches, which forward the traffic depending on the routing protocols or applications running in the SDN controller. In this context, the SDN controller interacts directly with the switches using standardized protocols, such as OpenFlow (17), strongly supported by the Open Networking Foundation (ONF) (18). This avoids the problem of having different vendor-based switches and their different interfaces and programming language. It is worth noting that SDN is not only a solution proposed in the research, but it has already been implemented in industries too. Although SDN has a centralized structure, due to resilience, clearly a distributed approach is also provided with different entities of the SDN controller. Typically, the distributed approach is also implemented when scaling the network up. A single entity might be affected by congestion when the number of nodes increases, and different entities are assigned different areas of the network to mitigate the high traffic load.

Complementary to SDN is the concept of Network Function Virtualization (NFV) (19). While SDN focuses to the softwarization of the routing protocols into centralized entity, NFV proposes the abstraction of those network functions, e.g., orchestration management functions, security, firewalls, that traditionally run in the physical devices. The paradigm is based on the simplification of the network devices by releasing these network functions and running them on servers on cloud. Clearly, this process makes the network devices easier which corresponds to less initial and maintenance costs. In addition, it increases the flexibility of the network to adapt faster to the need of the network. Traditionally, a specific network function required its device to be deployed. With NFV the network function is executed at the core of the network.

1.1.4 Satellite role in 6G networks

It is evident that more areas of daily life are always impacted and dependent on connectivity, health, politics, education (20). Historically, terrestrial and satellite networks were clearly two different and independent environments, with separate standardization process and results. The demand for ubiquitous and continuous connectivity calls for the need of a diversified network, where a Non-Terrestrial (NT) component is integrated with the terrestrial network in a seamless manner. This is driven also by the potential of the 6G market and applications where the satellite segment provides additional backhauling, capacity and

redundant paths. The technical reports from 3GPP (21; 22) address the relevance and benefits of investing into the combination of the NT element with terrestrial networks, while a recap of the activities and the project sponsored by the European Space Agency (ESA) with respect to this integration is reported in (23). On the one hand, NTN elements include satellites at different altitudes such as the low-earth orbit (LEO), medium-earth orbit (MEO) and geostationary orbit (GSO). Recently developed but lower altitude flying objects, such as high-altitude platform (HAP) are also considered part of NTN elements but will not be considered here as it goes outside the scope of the thesis.

1.2 Virtual Network Embedding

VNE has been an intensively studied NP-hard problem in the past decade. This thesis presents different solutions for VNE, based on the requirements of the considered scenario. For each chapter from 3 to 5, a specific brief and specific literature summary is presented to support the main motivation of the chapter. This section aims to provide a more general overview of the VNE. As briefly anticipated in section 1.1.2, VNE is the mathematical formulation for network slicing. VNE is the challenging allocation of any kind of service, which can be also seen as a subnetwork from the graph theory, in the physical or substrate network. Since it is the embedding of a subgraph into a larger graph, two processes are inevitably created, namely, node and link mapping. For example, Figure 1.4 shows two VNRs, with their respective resource requirements (node and link capacity). The node and link mappings, in mathematical terms, are functions.

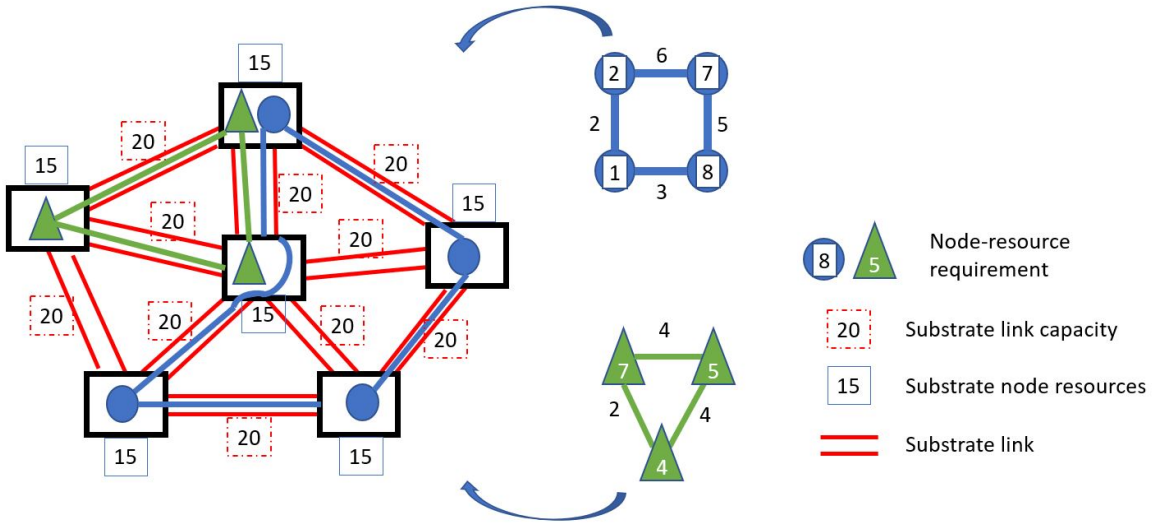


Figure 1.4: Virtual Network Embedding

Given N_S the set of substrate nodes, N_V the set of virtual nodes (nodes of the VNR), E_S the set of substrate links and E_V the set of virtual links, the node and link mapping functions M_N and M_L, respectively, are as follows:

M_N: $N_V \longrightarrow N_S$

M_L: $E_V \longrightarrow E_S$

Being VNE an NP-Hard problem, i.e., a problem whose resolution time exponentially increases with increasing network size, an unavoidable trade-off must always be considered in the solution. This means that an exact solution can be found in a reasonable time for very small networks. On the contrary, mainly heuristics and meta-heuristic solutions are proposed to find a local optimal solution in a reduced amount of time even for large-scale networks. In (24), authors analyze the complexity of VNE, investigating different variants of the problem, in terms of relaxed constraints. The resource constraints, like the node and link capacities, node location, latency and routing restrictions have been relaxed alternatively in different variants. Some combinations of them are studied, e.g., VNE with only integer constraints of node and link capacities, and with relaxed constraints on node location and latency requirement. However, all the investigated scenarios have shown the NP-completeness. The complexity of the problem has opened the way for researchers to analyze VNE from different perspectives and with different methods. Indeed, depending on what has the highest priority between the computation time and the optimality, different approaches can be investigated.

In the literature, several differentiations in VNE approaches are proposed. Authors in (25) describes some of them as follows:

- Static/Dynamic depending on the possibility or not to reallocate resources, whenever a new VNR comes in the network, for a better exploitation of the substrate network;
- Centralized/Distributed referring to the role of the entity, in the network, in charge of solving the VNE;
- Concise/Redundant depending on the strict or redundant assignment of resources when the mapping is performed.

Since all these features are considered independent, every mapping algorithm can be developed following any combination of the six above-mentioned features. Consumable resources are released when the VNR is no longer active. On the other hand, there are static resources, such as the loss probability or delay of a link, which are independent of the number of mappings, even though they might still vary over time. Every VNE algorithm is formulated with constraints and objectives. Typically the objectives are a minimization function which targets either the average use of the substrate resources (nodes and links), the ratio cost/revenue or the packet loss. VNE algorithms are evaluated on some common evaluation metrics or KPIs as listed below:

- The *acceptance ratio* which is the ratio between the amount of accepted and embedded VNRs over the total amount of received VNRs;
- The *revenue*(resources required and allocated to the VNR)/*cost*(physical resources) spent from the InP which should be able to exploit its physical resources in the best possible way;
- Quality of Service metrics, like delay, jitter, throughput, network element utilization, should respect the QoS of the VNR, after the embedding is done;
- The reliability of the embeddings with some resilience metrics like backup-resources, path redundancy, migrations;
- Further metrics like the execution time of the algorithm, the needed signaling and the number of active substrate nodes.

In addition, VNE solutions differentiate one each other for whether the node and link mapping are computed independently, typically in a sequential manner starting from node mapping and then the link mapping, or in a combined manner, one stage approach. The last option is usually more complex and optimal because it finds a coordinated solution.

1.3 Motivation

This section aims at summarizing the main motivations of this thesis. Firstly, we underline the reason why it is important to study the VNE for the novel networks, i.e., satellite-terrestrial integrated networks. It is evident that the study of VNE algorithms has improved the way physical resources are used. With the change of telecomunications standards from 4G to 5G and nowadays from 5G to 6G, the request of resources has drastically changed over the years, both from type and from load perspectives. This had an impact on the use of the substrate resources that necessitate always improved optimization levels. As typically the study of a VNE algorithm matches the current requirement and network features at the time it is proposed, the development of new VNE algorithms is unavoidable to match the fast progress and change of the telecom networks. Following this trend, this thesis analyzes some major and recent novelties in the physical network and proposes, accordingly, novel formulations for VNE algorithms. For example, with the development of ISLs, the satellite segment became less trivial and increased the number of available network paths. This triggered the study of a VNE which exploits the ISLs to minimize the traffic handovers, see chapter 3. With the high traffic load and physical resources being 100% used, it becomes relevant to deeply analyze the embedded traffic, and based on prediction models and/or statistical description, to exploit its non-stationarity and assign the unused resources to other traffic requests, and, consequently, improve the overall utilization of resources, see Chapter 4. Finally, recent advances in LEO constellations and their integration with the terrestrial segment created a very dynamic network that makes the development of a network virtualization paradigm, such as network slicing, difficult to implement. It is necessary to study its implementation in this type of integrated networks, and provide a solution that is able to react to fast changes and congestion scenarios at the same time, see chapter 5. To conclude the motivation chapter, the testbed proposed in this thesis makes a significant contribution to the literature. The literature for VNE has proved to lack testbed with emulated networks to test the algorithms

and, eventually, create feedback loops to give rewards to the algorithm and real-time modify the embedding. This is why the algorithms proposed here, except the last one due to time constraints, are developed and tested within the testbed with real traffic.

1.4 Contributions

The total contribution of this PhD thesis is distributed among chapters 2, 3, 4 and 5 and is as listed below:

1. **Chapter 2:** describes the different versions of the testbed until the final one, with a detailed description of each feature;

2. **Chapter 3:** Virtual Network Embedding reduced to link mapping for mapping in SES O3b MEO links. In this scenario, the main point was to minimize the number of traffic handovers over time and show the first integration with the testbed;

3. **Chapter 4 :** In this chapter, we show that the virtual network embedding is improved by the idea of collecting traffic and run an optimization problem that not only optimizes the embedding in terms of routing but also in terms of datarate assignment;

4. **Chapter 5:** This chapter presents the full VNE (node + link mapping) for satellite-terrestrial networks (GEO-LEO) with constant monitoring to perform handovers even in a reactive manner (e.g., congestion) with minimization of the reconfiguration cost;

5. **Chapter 6:** Conclusive remarks and future research directions.

Chapter 2

MIRSAT: SDN-based testbed

This chapter provides the detailed description of the testbed we built, starting from the literature review, the main motivations, the description and the reason for the tools that have been involved, the features, capabilities and the results.

2.1 Related Works

There are several implementations in the literature that propose solutions for SDN-based testbed, considering satellite or integrated satellite-terrestrial networks at the physical layer. In this section, we analyze the main ones to highlight the motivations that drove us toward the realization of our built-in SDN-based testbed for integrated multi-layered satellite-terrestrial networks. In (26), authors describe a software-defined satellite network, where a preliminary analysis is proposed. The control plane, namely SDN controller, is proposed to run on GEO satellites that have a broader view of the network below (lower orbit satellite and terrestrial elements). Although this is a valuable proposal, it was still left to premilinary analysis without providing a platform for it. In (27) authors propose a networking scheme for an SDN-based layered satellite network (GEO-MEO-LEO), specifically oriented to an operationally responsive space system. Even in this case, no prototype is provided, but only a theoretical network model to be implemented in an SDN architecture. Another preliminary analysis is proposed in (28), with an expanded layered network, where not only GEO, MEO and LEO satellites are involved, but also HAP and lower altitude aerial platform. The proposed network model foresees the control plane installed in terrestrial nodes, HAPs and GEO satellites while the data plane composed by the rest of the nodes. Even in this case, a

Table 2.1: SDN-based solutions for satellite networks

Architecture	Data plane elements	Implementation	Multi-layered
OpenSAN (26)	GEO, MEO, LEO	preliminary analysis	yes
SDN-SatArc (27)	GEO-MEO-LEO	preliminary analysis	yes
MLSTIN (28)	GEO, MEO, LEO, HAP, terr nodes	preliminary analysis	yes
SDSN (29)	GEO, LEO & terr nodes	prototype	yes
SERvICE (30)	GEO-MEO-LEO-terr nodes	prototype	yes
HetNet (31)	satellites & terr elements	prototype	no
MIRSAT	GEO, LEO and terrestrial	prototype	yes

prototype is not provided. Authors in (29) propose a software-defined network for a layered GEO-LEO infrastructure. In this scenario, the authors focus on the handover procedure, initiated on the Signal-to-Noise Ratio (SNR) level. The SDN controller is hosted on ground centers and the benefit is shown via the use of the GEO satellite to install the forwarding rules in the LEO satellites in its field of view. In this work, the scenario is emulated in Mininet but the SDN controller is not specified. While it is a first example of an implementation, it is dependent on the scenario analyzed, i.e., handover mechanism and, hence, it lacks of flexibility to be used with different scenarios and algorithms. In (30), a layered infrastructure with LEO-MEO-GEO satellites is proposed. SDN is in combination with NFV to provide a more complex end-user service. Authors implement network protocols from LEO access point to the exit point with the intent of improving the user experience. The satellite simulation is realistic due to the use of STK and there is a QoS-based analysis (delay, capacity and packet loss) of the user experience, however the architecture does not provide any traffic-related feedback between the SDN controller and the data plane network, which is essential to manage real-time scenarios and improve the resource allocation. Following the same logic, (31) proposes an SDN-based framework for satcom with implementation of NFV throughout the network. The system provides networking capabilities through programmable switches. We summarize the literature review in Table 2.1.

2.2 Motivation

After the literature review, in this section, we highlight the motivations for MIRSAT. We specifically describe the reason why the testbed has become relevant in the field of VNE and the benefit of providing an SDN-based testbed. Then, the motivations for the choice of the

several tools involved in the testbed are provided.

This thesis discusses different proposals for an embedding algorithm, named VNE. These studies analyze the complexity of an NP-Hard problem and show the mathematical formulations based on the specific objective each chapter wants to achieve. In any case, each implementation shares the final target, i.e., optimizing the physical resources allocation. To be efficient but at the same time not too far from the reality, it is also relevant to provide real use-cases or similar, which can show the reader how the proposed algorithm would be used in a real context, for instance, from a service provider point of view. This is an aspect that in the research is not always considered, probably due to the fact that it may be time consuming. However, it is clearly the best way for researchers to bring mathematical optimizations closer to the industry. Furthermore, the testbed is also useful when there are data-driven scenarios and a resource allocation problem must be data-driven to be applicable in the novel networks which are very dynamics. This means that a testbed, which constantly monitors the status of the network and feeds back the algorithm, is an essential add-on component to the development of any mapping algorithm.

In section 1.1.3, we provided an initial introduction to the concepts of SDN and NFV. We underlined that these two novel technologies have become more important in the current transport networks as they proved to be the main enablers for the network virtualization. In this section, we will focus mainly on SDN as it is the main player of our built-in testbed. SDN is one of the most successful enablers in field of VNE because, thanks to its feature of providing a full description of the substrate network, it supports the centralized decision of how and where to allocate an incoming VNR. In fact, the VNE algorithm is generally computed in a centralized way. From an SDN perspective, this makes it ideal to be run within the framework of the SDN controller. In addition, the SDN controller is implemented to keep track of any change in the physical network, both for connectivity, e.g. variation of the links during time, and for current throughput. This allows to build a constant monitoring tool which is capable of reactively responding to the status of the physical network and take actions in low time. Finally, the strong advance in the SDN implementations in the literature, makes the life easier because of the vast multitude of choices. Among them, easy ways to use the implemented controllers via well-implemented manuals and interfaces, such as REST interfaces, help and further motivate the use of this technology.

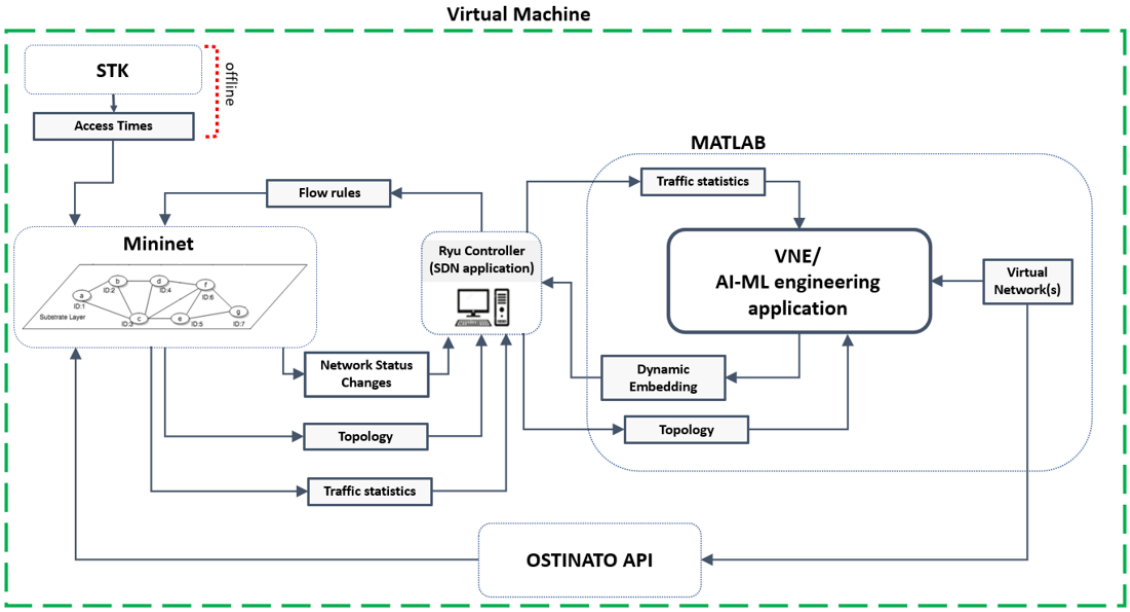


Figure 2.1: MIRSAT testbed - Final version

2.3 Tools

The testbed in its final version, Figure 2.1, is composed by five main elements as follows:

- **Ryu**: the SDN controller;
- **Mininet**: the network emulator;
- **Matlab**: the environment where the VNE algorithm runs;
- **Ostinato**: the traffic generator;
- **STK**: the satellite simulator.

In the following, we describe each main element, its role within the testbed and the reason why it has been chosen. Then, the different versions of MIRSAT are presented.

2.3.1 RYU - The SDN Controller

The SDN Controller is the main brain of any SDN-based network. While chosing the controller, different criteria should be considered, such as the advancement of the currently available versions, the community of users (the larger the better, to increase the chance of finding replies to problems that may raise during installations or implementations), the programming

language, the communication protocol with the substrate network and if it is open-source. We selected RYU (32) because it is simple to use, open-source, python-based and with already a strong development behind which simplifies the problem solving, especially during installation phase, and the writing phase when new features are required. Ryu provides a REST interface which we intensively used in the testbed to communicate in both directions from and to the VNE algorithm, running in Matlab. As can be seen from the structure of the testbed in Figure 2.1, RYU controller has the main role. In fact, on the left side of the figure, the interaction with the network emulator is based on (1) the constant monitoring of the physical network, i.e., checks the connectivity of the links (*Topology, Network Status Changes)* and the throughput that is being exchanged over each link (*Traffic statistics*) and (2) the installation of the entries in the forwarding table of each programmable switch (*"Flow Rules"*) and of rate limiters (*Dynamic Rate Limiters*) to limit the maximum throughput that a specific traffic demand can use. The right side is instead the communication with the VNE algorithm. In this context, RYU sends the traffic statistics and the real-time topology, which are asked by the VNE algorithm via GET request. After every new successful computation of a new mapping, the VNE algorithm sends the results (vector of intermediate nodes from source to destination) to the controller, which translates the vector into flow rules in each involved intermediate node.

2.3.2 Mininet - The network emulator

The network emulator plays another important role in the testbed. The traffic that is generated normally in simulations, now it has to go through the substrate network provided by the network emulator. Therefore, it has to be compliant with the available hardware in terms of resources's demand and it needs to be easily deployable, flexible and well advanced in the development, for the same reasons we mentioned in the previous section related to the SDN controller. It was crucial to select a network emulator with the dynamic capability of turning down and up links during time as the substrate network since the beginning was supposed to be a combination of terrestrial and NGSO links. In this case, Mininet (33) is the ideal network emulator because is open-source, with a set of available features which is not very large but sufficient considering the scope of our testbed. In addition, a large community makes use of it and it provides the most typical version of programmable switch, OpenVSwitch (34), which communicates via OpenFlow (17) to the SDN controller. Lastly, Mininet has easy-to-use CLI

and GUI for different types of users and needs and it is easily scalable if new switches or links needs to be deployed. Mininet is the passive entity of the testbed because the main activity performed is the execution of the code where the timely description of the links and their capacity is provided. It is worth underlining that each network's node is emulated in Mininet as either a programmable switch or a host. To be more specific, we use a switch to emulate a satellite or a terrestrial node which acts as a forwarder. We use a host to emulate the real traffic generator agent, for instance this can be a mobile phone, a plane, a ship, a server.

2.3.3 Matlab - The VNE algorithm

The third element is the environment where the algorithm runs. The choice for this is strictly linked to the computational capabilities required by the optimization algorithm. In this case, VNE is a NP-Hard problem which typically requires good solvers to be solved in considerably low time. Matlab (35), developed by Mathworks, is one of the most common and known computing platform, used by engineers in many different fields when it comes to develope and analyze model, data and algorithms. It is convenient to use Matlab for VNE, because not only provides a large set of solvers, such as CVX, for convex optimization, but it also offers a framework where many different functions can be managed, such as the creation of traffic demands, their lifecyle, the retrieval of the traffic statistics and the analysis of the final results. While these functions may run within the SDN controller, it is efficient to unload them to Matlab because this increases the flexibility to use the testbed for different types of algorithms, such as AI-based mapping algorithm which requires the input-output logic with an emulated network. To conclude and to summarize, the role of Matlab is (1) to manage the creation of the traffic demands and send the details (source and destination, timely datarate and load of data to be transmitted) to the traffic generator, which will instatiate the traffic accordingly, (2) to host and execute the VNE algorithm and (3) to input/output the communication with the SDN controller.

2.3.4 Ostinato - the traffic generator

In the majority of the VNE implementations, Matlab manages everything, including the traffic generation. As we build a testbed to emulate a real scenario, where streams of traffic flow in the network because users requires connectivity to specific servers or applications, it is relevant to use a traffic generator which can emulate this scenario as close as possible

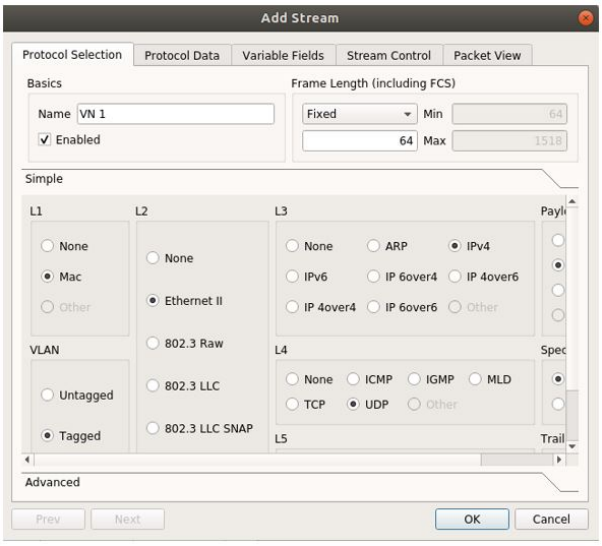


Figure 2.2: Ostinato GUI 1

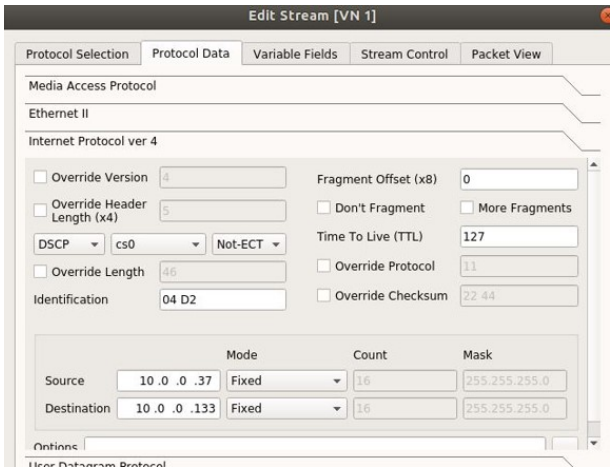


Figure 2.3: Ostinato GUI 2

to the reality. It is useful that the generator can characterize the traffic based on several features at the different levels of the protocol stack ISO/OSI. Ostinato (36), as shown in Figures 2.2 and 2.3, allows to differentiate the traffic with criteria such as the VLAN tag, the MAC/IP and the TCP or UDP details. In addition, since Ostinato works with stream of traffics, which means that an amount of data is set to be transmitted at the beginning with a certain datarate, it allows to emulate a non-stationary traffic by creating a sequence of streams, as shown in Figure 2.4. In this example, a traffic is emulated with a change of datarate and dataload in four different streams. The Ostinato GUI shows the real-time transmitted (highlighted in the red dotted box) and received throughput.

Once the input from Matlab arrives, Ostinato creates and instantiates the streams of

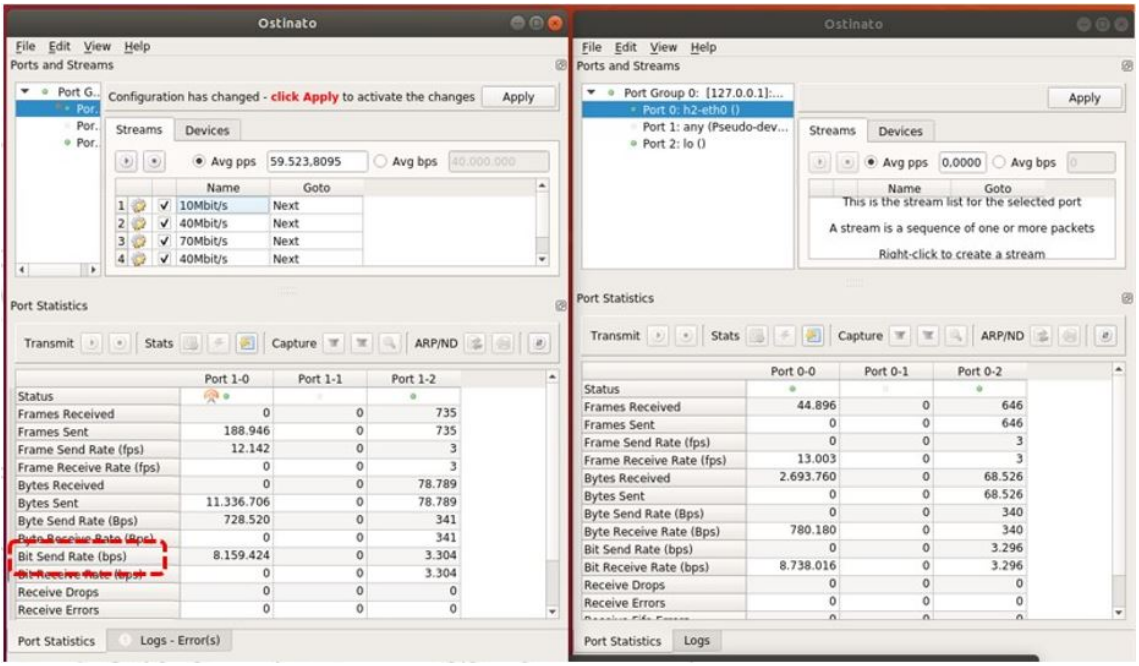


Figure 2.4: Ostinato GUI 3

traffic and sends them to the host source. An Ostinato instance is installed in every host in Mininet to facilitate the communication between the two tools.

2.3.5 STK - The Satellite simulator

Most of the work included in this thesis is based on satellite networks. When dealing with constellation of satellites, one can search online for the structure and the details of it but the most precise manner is to simulate the constellation of interest in a proper simulator. STK (37) is a well-known satellite simulator which simulates with detailed feature any type of constellation, with 3-D visualization. The user needs to include (1) the details of the satellites, i.e., the orbit, typically stored in the STK databases, (2) the location of the ground terminals and (3) the location of the user terminals. Once this is fixed, STK provides the time description of each connection between any node (satellite-satellite, satellite-user terminal, satellite-ground station) involved in the network. This is an important step to be performed because from the timeline description of the network, we can replicate it in the testbed. In fact, once the scenario is set, we perform a simulation in the selected interval of time and we download an excel sheet with the duration of each connection. From this, a Matlab function converts the data into Mininet code to be able to replicate the exact behavior of the dynamic links over time when running the network in Mininet.

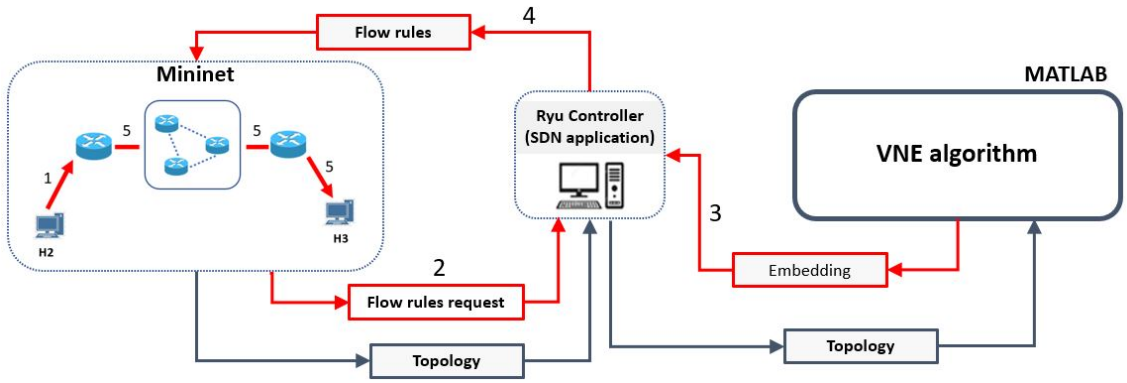


Figure 2.5: MIRSAT - Version 1

2.4 Basic capabilities - Versions' History

This section has the intention to go through the main capabilities of the testbed which is also a sequential recap of the different versions of the testbed, starting with the simplest and easiest one till the final version that is running today.

2.4.1 Version 1 - E2E Connectivity

The first and easiest version of the testbed (see Figure 2.5) is providing the connectivity between two end-points, emulated in Mininet as two hosts. The main functionalities here are as follows:

- instantiation of the traffic request, a ping request is initially used;
- the request is then transferred to the Matlab algorithm which computes the path;
- the SDN controller fill the forwarding tables of all the switches from the host that generates the traffic until the

In Figure 2.5 the steps are explained in details and in order.

1. The ping session is generated betweena source and a destination host (pair of IP addresses of hosts h2-h3). The first generated packet arrives to the first switch connected to the host h2;
2. The switch triggers the flow rules request, which are sent to the SDN controller;

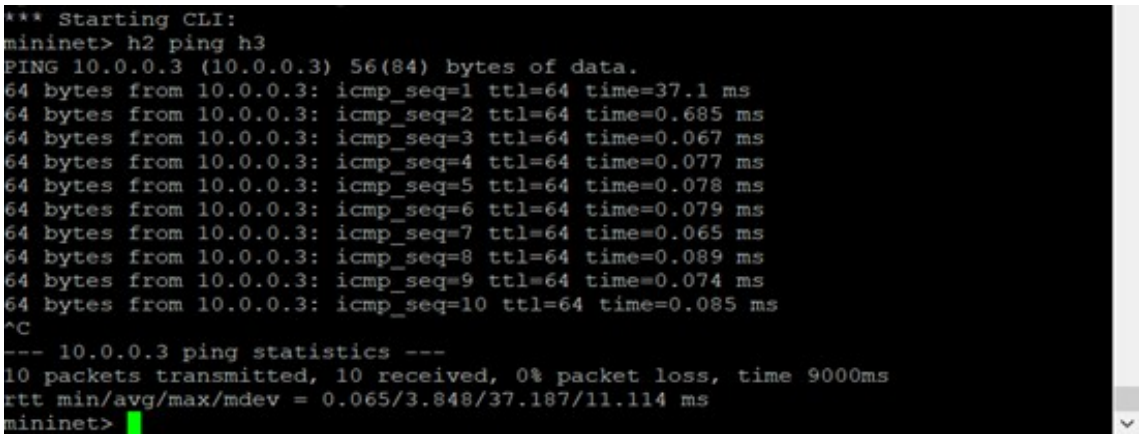


Figure 2.6: Check connectivity

3. The SDN controller recognizes the identifier of the VNR (via a datasets where IP-pairs correspond to a specific VNR) and reads the output (embedding path) of the VNE algorithm for that VNR;

4. The SDN controller installs, in all switches between the source and destination, the necessary flow rules, in the forwarding tables, to provide connectivity;

5. the traffic flows between the pair of hosts.

The connectivity is proved via a ping test, as shown in Figure 2.6. From the ping results, it can be noticed that only the first packet is impacted by the delay of the above mentioned steps (1-4). This means a delayed Round-Trip-Time (RTT).

2.4.2 Version 2 - Dynamic links

After checking the connectivity for a fixed configuration, in this subsection we prove that Mininet emulates even dynamic links. Each link in Mininet can be set up and down during time, to simulate a dynamic link. This applies well to the satellite context, where in NGSO orbits, links are almost always dynamic with an active period when the satellite is in LoS with the terrestrial antennas (either gateway or user terminal). In addition to the dynamicity, we introduced the delay in every satellite link. Even in this case, we take a simple scenario, as shown in the left picture of Figure 2.7 to initially show how the satellite features is emulated.

To run this test and check the traffic connectivity, *iperf* was used and the results are shown in the right picture of Figure 2.7. The simulation simply runs a temporal satellite connection between two terrestrial hubs (in this example we do not enter into the details of the nodes for

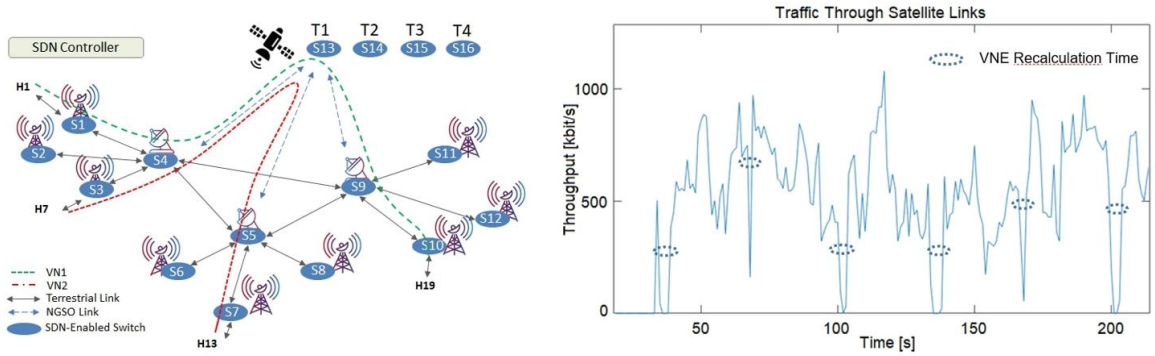
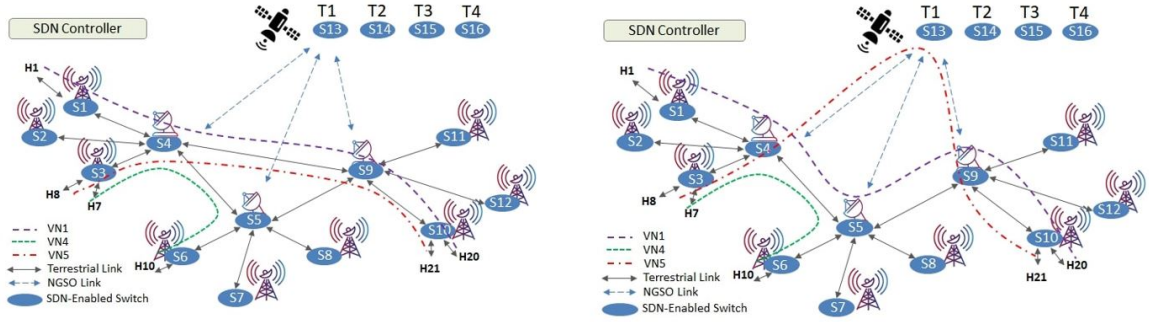


Figure 2.7: Simplified satellite scenario - 1

the sake of simplicity). Two virtual networks are simulated, VN 1 which is routed through switch S4-sat-S9, and the VN 2, routed through switch S4-sat-S5. The switches from S13 to S16 represents 4 different NGSO satellites, which cover the interested area sequentially during the time. In other words, the first satellite in LoS will be represented by the switch S13, then S14, and so on and so forth. In the simulation, this process is done automatically with a pre-set timer in Mininet to 30 seconds. From Figure 2.7 it can be noticed that the throuhput decreases significantly for some small periods of time. In fact, during the change of the topology (every 30 seconds) some time is needed for:

- the VNE algorithm to realize that the topology has changed, thanks to the constant (almost every time) process of topology retrieval, executed by the SDN controller, see Figure 2.5;
- compute the new path;
- send the output to the SDN controller which applies then the new flow rules to the switches.

It can be computed that this process typically requires few seconds. For these results, iperf was used to check the variance of the throughput over time in a dynamic scenario. As we added the latency on the satellite link, and *iperf* does not provide information on that, it is useful to use ping session to check that the latency are correctly emulated. For this reason, we simulate a second scenario, as illustrated in Figure 2.8, where three virtual network requests are simulated. In this scenario, as we consider MEO satellites, we set the delays equal to 27 ms (i.e., RTT equal to $\sim$ 108 ms). We take the chance not only to show the delay, but also



(a) *Mapping before failure* (b) *Mapping after failure*

Figure 2.8: Simplified satellite scenario - 2

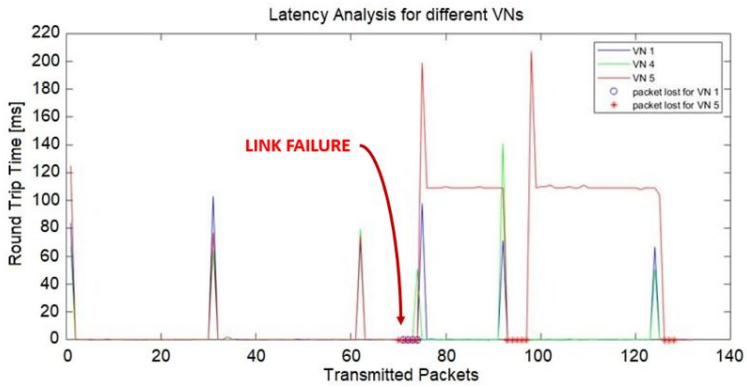


Figure 2.9: Latency results

to show that the system is prompt to recover from a random and unpredicted failure. The following scenario considers a terrestrial link failure, after $\sim$ 70 seconds from the beginning of the simulation. Initially, the three VNRs are embedded over the terrestrial links. Then, after the failure, the VNE algorithm computes new paths only for the affected services (VN4 and VN5). VN5 (red line in Figure 2.8(b)) is forwarded to the satellite link while VN4 is forwarded to another terrestrial link.

Since the ping session gives latencies, for each transmitted and received packet, as output, Figure 2.9 depicts on the x-axis the transmitted packets (which also correspond to the time in seconds considering that every second 1 packet is sent) and on the y-axis the experienced latency. It can be noticed that the terrestrial embedded VNRs (VN1 and VN4) experience always $\sim$ 1 ms of latency, despite the peaks when topology changes happen.

On the contrary, VN5 (red line) experiences the terrestrial latency before the failure, and a MEO latency after the failure.

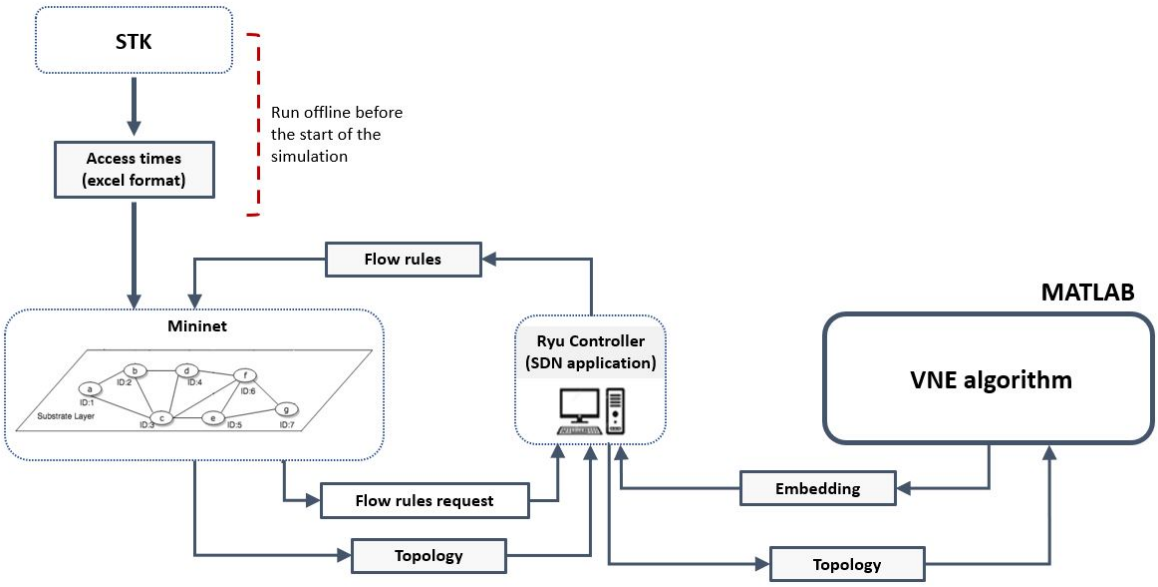
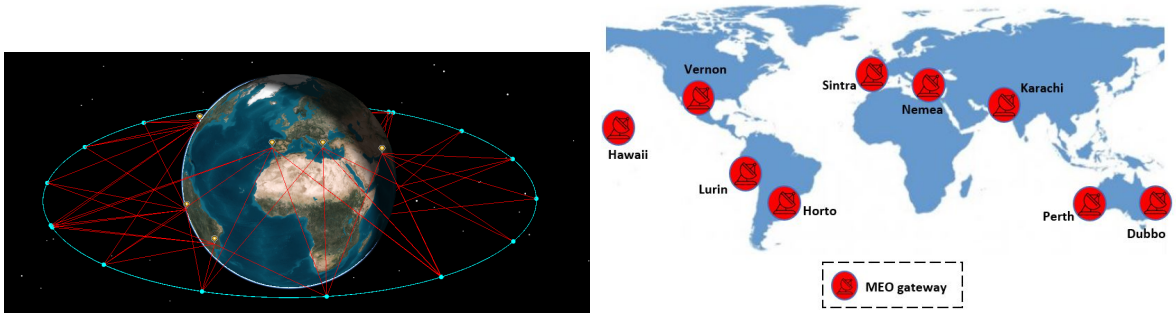


Figure 2.10: MIRSAT - Version 3

2.5 Version 3 - Real satellite constellation

The third version of MIRSAT aims at giving more real features to the testbed. In the second version, a satellite link is simulated but has not practical correspondence in the reality. For this reason, the third version of the testbed introduces the real satellite emulator STK, see Figure 2.10.

For the first experiment, we took the SES O3b constellation (currently being enhanced with O3bmpower (38)), a MEO constellation of 20 satellites at an altitude of around 8000 km.



(a) O3b constellation (b) SES Network map - MEO Gateways

Figure 2.11: STK scenario

Figure 2.11a shows the simulated fleet of satellites and their link to the terrestrial MEO

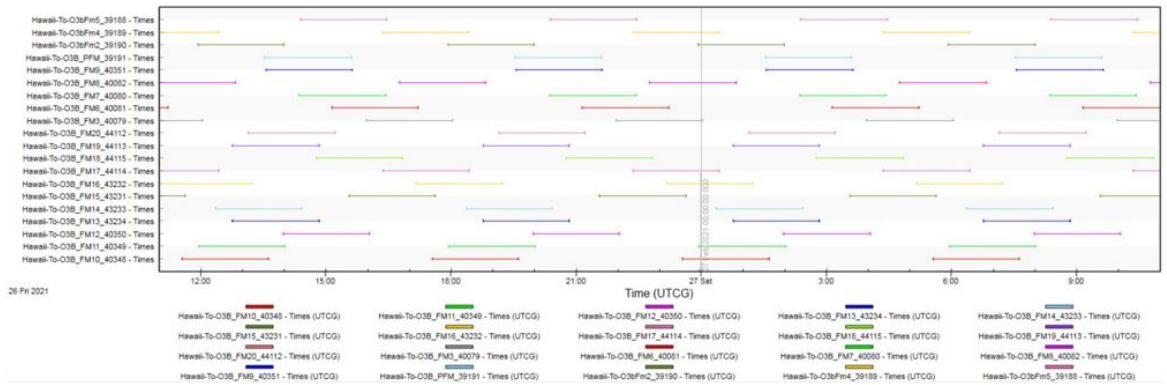


Figure 2.12: Access time scheme of Hawaii gateway to the 20 O3b satellites over 24 hours

Gateways. Every satellite has more than one link, which means that it is in visibility of more than one gateway (GW). In Figure 2.11b, the map of the MEO GWs is shown. As mentioned in Section 2.3.5, the STK is running offline and upfront. This preliminary process is required because the output of STK are used to generate the timely description of the network, and to prepare the code of Mininet which runs as soon as the simulation starts. All the needed information is contained in an excel file named "Access Times" which can be download from STK. The access time sheet prints out the duration of each link that is selected. In particular, in our scenario, we need to download one excel file for every terrestrial GW which describes the duration of each link between the GW and all other satellites. Clearly, links are discontinous because we are talking about NGSO links. An example is shown in Figure 2.12.

The conversion process from excel files to Mininet code is automatically executed by a Matlab function. This is convenient because even with a "small" network like O3b, the number of changes are already too high to be handled manually.

2.6 Version 4 - Dynamic Rate Limiters

Although all previous versions of MIRSAT used the most common and well-known Open-VSwitch (34) as a programmable switch, this version uses a different type of OpenFlow enabled switch, called CpQD (39). The reason for this is that the OpenVswitch does not provide the main feature we described in this section, the rate limiters. One of the main tasks of an SDN controller is to be able to assign link capacity to different customers based on their requirements and requests. This translates into a subdivision of the capacity and

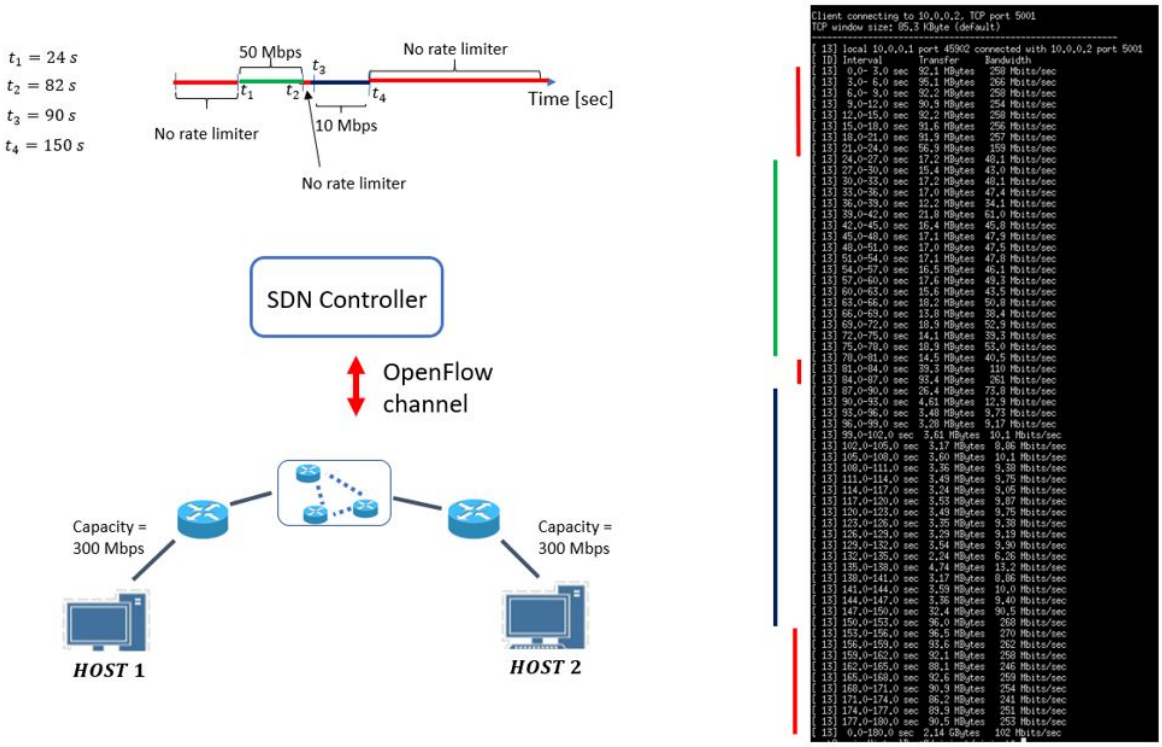


Figure 2.13: Rate Limiters application

to an assignment to each customer, i.e., VNR, a maximum datarate that can be used. This means that if the VNR sends traffic with higher throughput, the throughput will be blocked at the maximum allowed speed. This is the definition of a rate limiter. It turned out that only CPqD works with the rate limiters.

In the following, a simple example of application of rate limiters is explained in Figure 2.13. Host 1 (h1) and host 2 (h2) belong to the same VNR and exchange traffic. On the top of the picture, the timeline shows the intended limited rate, i.e., for the first 24 seconds no rate limiters are implemented, then the rate is limited at 50 Mbps. After 58 seconds, rate limiters are removed for few seconds and are set again, but limiting the rate at 10 Mbps to show their dynamicity. Finally, after 60 seconds, no rate limiters again. We note that, when there are no rate limiters, we expect the traffic to use the full capacity of the link (in this scenario because the link is empty and there is no other traffic passing through it). The capacity of the links is set to 300 Mbps. The rate is tested with iperf from h1 to h2. On the right side of Figure 2.13, the rate limiters are reflected in the iperf test results. It is worth noting that the CPqD switch does not currently support the dynamicity of the link, i.e., the real-time activation and de-activation of the link. In fact, while using CpQD switch,

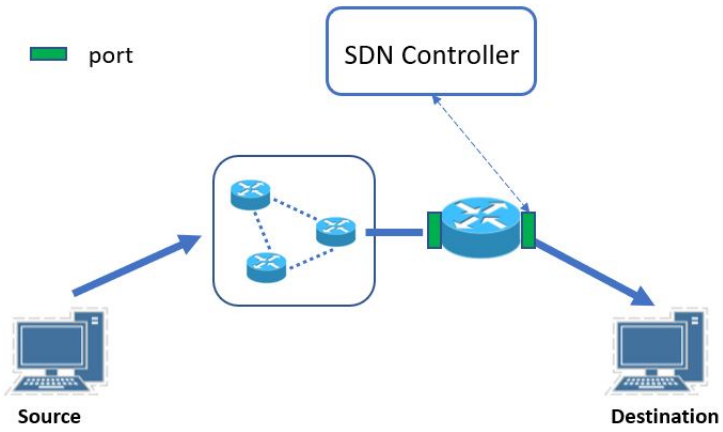


Figure 2.14: Computation of real-time throughput

once the link is de-activated, it does not activate anymore so a satellite network can not be emulated. For this reason, from now on, we drop the use of CpQD and we use the traditional OpenVswitch. A further version of the testbed with a solution for this issue is planned.

2.7 Version 5 - Traffic statistics collection and traffic generator

The fifth and final version of the testbed, see Figure 2.1, includes the use of traffic statistics collection and Ostinato traffic generator to further improve its application to real scenarios. Among the features of the testbed, the collection of traffic statistics is relevant to close the loop of any generic resource allocation algorithm. In fact, the algorithm is executed based on the initial resources available, but then, the real use of the resources is necessary to feed back the algorithm and, eventually, drive the decision toward a better exploitation of the last ones. We note that AI/ML algorithms can strongly benefit from these features as some of them, such as Reinforcement Learning, are strongly based on feedbacks.

For this purpose, we provided the SDN controller with this feature, and the mechanism is explained in Figure 2.14. A source sends throughput with a datarate to the destination host. A set of intermediate switches has the installed flow rules to allow communication between them. Among the parameters that OpenFlow provides for each flow rule, "*Received Bytes*" is a counter that continuously increases with the total bytes of data that matched the flow rule. As the SDN controller can retrieve at any time any flow rules from every switch under control, the real-time throughput is simply computed as follows:

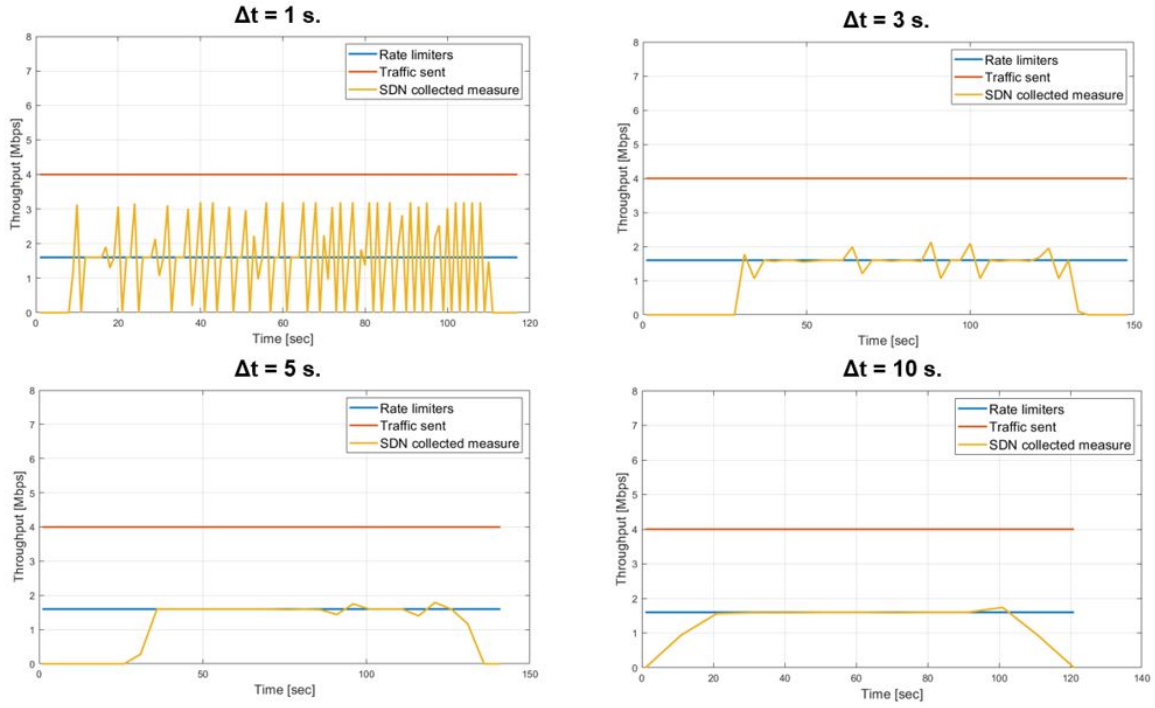


Figure 2.15: Δt comparison

$$Datarate(t) = \frac{ReceivedBytes(t) - ReceivedBytes(t - \Delta t)}{\Delta t} \tag{2.1}$$

Thus, for every interval of time Δt, the SDN controller computes the datarate. Clearly, depending on the Δt, the results may vary considerably. For instance, if Δt is too long, then the fluctuations of the traffic are not properly reflected in the obtained throughput, but instead only an average will be computed. If Δt is too small, it can give unstable results. An evaluation of the impact of Δt is provided in Figure 2.15.

We tested it while sending a traffic at constant rate of 4 Mbps and being limited at 1.6 Mbps. It can be seen that when $\Delta t = 1$ s, the received data fluctuates too much (even though the average is still the correct value). This tells that 1 s. is a too small value. Already with $\Delta t = 3$ s., the results are more stable with fewer fluctuations. As expected, the most stable results are obtained when $\Delta t \geq 5$ s. This was just a simple evaluation with constant traffic, but clearly the evaluation of the right Δt must match the behavior of the traffic. For example, a granularity of 10 s. might be very large for some applications such as emergency services, which are typically highly non-stationary.

The last feature to be described is the introduction of the traffic generator Ostinato,

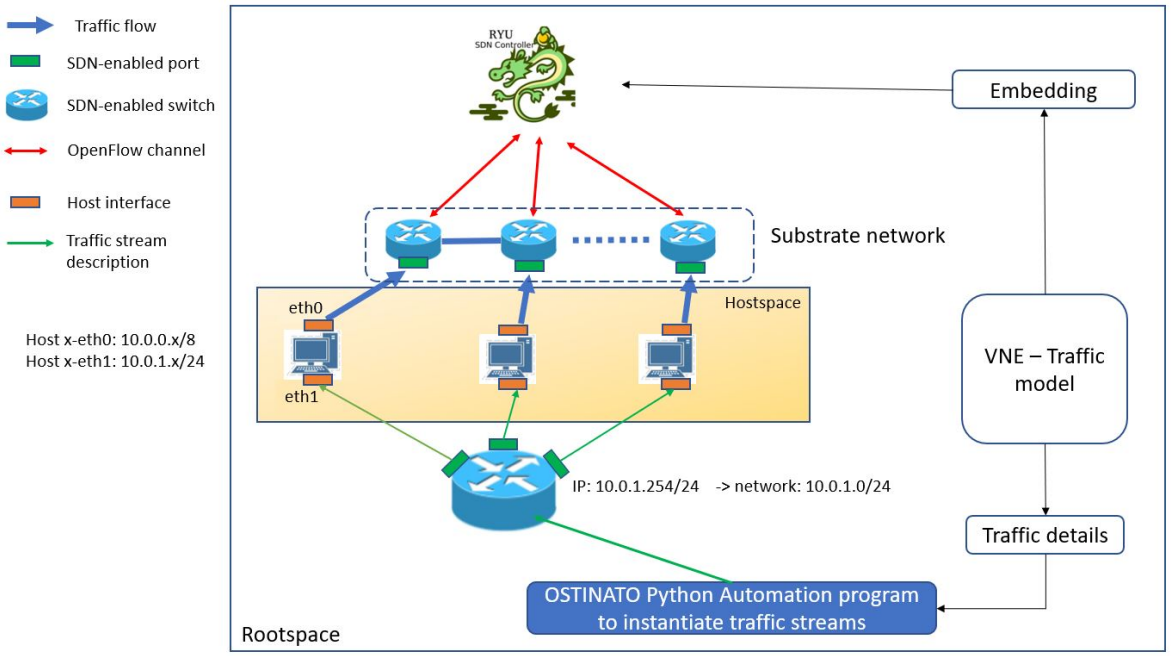


Figure 2.16: Ostinato API architecture

see Section 2.3.4. Before using Ostinato, as also shown in the figures above, every test was performed with either ping or iperf. These well-known tools are useful for testing connectivity and datarate, respectively. However, one added value of the testbed is to simulate real scenarios, where traffic varies in datarate during time. Theoretically, iperf allows to set the datarate of the traffic and automatic scripts can be created to simulate a communication with different datarates over time. However, Ostinato provides easier user-friendly functions to set the stream of traffic with different traffic details from layer 2 to layer 4 of the ISO/OSI protocol stack. This allows to differentiate not only VNRs, but also traffic within each VNR. This is a relevant feature when the testbed is used for network slicing, where each VNR can be seen as an entire slice. At this point, different internal requests may be generated and treated differently.

In addition to this, Ostinato developers provide an Application Programming Interface (API) which facilitates the interfaction between the tool generating the traffic (process of definiting details of each stream of traffic) and the Ostinato which effectively implements the traffic and sends it in the emulated network. This process is explained in Figure 2.16.

As explained in (40), the traffic is generated by a drone application that runs on every Mininet host when the network is started. Typically, without Ostinato, every host is equipped with one link, i.e., one interface, connected to the switch to get the connectivity. In this case,

Table 2.2: Ostinato API test - traffic details

Vlan ID	type of traffic	Datarate	Transmitted data
1	In-Flight video streaming	25 kbps	2.5 MB
2	In-Flight text messaging	1 kbps	1 KB
3	Autonomous ship navigation	10 kbps	10 KB
4	Vehicle Collision Avoidance	10 kbps	100 KB
5	Manufacturer updates	5 kbps	375 KB

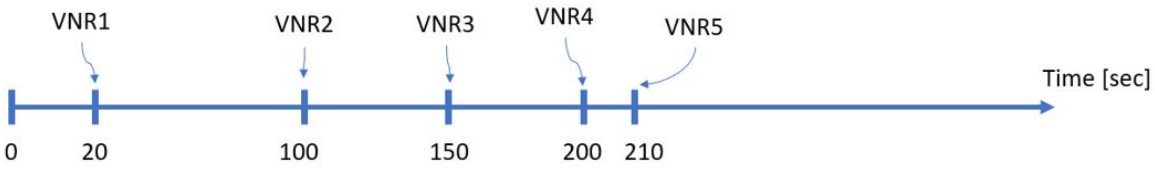


Figure 2.17: Traffic generation timeline

we equip every host with a second interface, connected to an external switch (external with respect to the main network), with an IP address setting, that is used to communicate with the drone application in the host from which we want to generate the traffic. This system has been implemented to avoid interference between management traffic and useful traffic. To recap, the traffic model of each VNR is defined in the algorithm, i.e., description of the datarate over time, the amount of data to be transmitted, the vlan id, and layer 3 and layer 4 details. Then, these details are passed with a python program to the drone of every host source. To test the latest two features, we run a simple scenario with five different types of traffic demands, as explained in table 2.2.

The traffic is generated not simultaneously, but following the timeline shown in Figure 2.17. After this step, we check with the SDN controller that the traffic is generated accordingly.

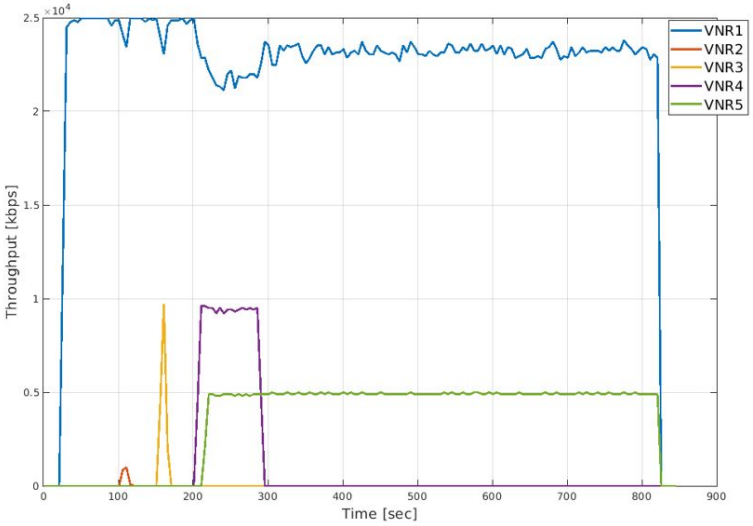


Figure 2.18: SDN collected traffic

Chapter 3

Virtual Network Embedding for NGSO Systems: Algorithmic Solution and SDN-Testbed Validation

3.1 Introduction

In chapter 1, VNE and its general motivation are discussed. The following chapter (1) discusses the application of VNE to NGSO networks, which is proven to be a hard optimization problem (41), (2) proposes an innovative solution to handle the number of handovers during the lifetime of a VNR and (3) describes the test of the algorithm in a SDN-enabled network. Due to the heterogeneous nature of novel virtual networks, the dynamicity of the substrate network introduces an additional layer of complexity which needs network slicing algorithms aim to provide intelligent mappings, with the intent to satisfy and guarantee to each of them a pre-defined QoS agreement.

NGSO networks, comprising the LEO, High Elliptical Orbit (HEO) and MEO constellations, can satisfy relevant common QoS requirements from current networks such as global high-speed coverage with a reduced latency, if compared to GSO networks (42), (43). NGSO satellite networks, due to their dynamic nature, especially LEO constellations, make the development of an efficient VNE algorithm challenging to deal with frequent network-topology

variations. In general, the lower the altitude of the constellations, the more often the topology changes over a defined period of time. This is due to the fact that the Line-of-Sight (LoS) visibility between satellite and gateway has a limited temporal duration. The time-dependent LoS visibility between a gateway and a satellite is in general priori known and, as such, does not require any statistical prediction. This is not the case for link failure/degradation, for example, which usually are simulated and predicted through a statistical description.

In this context, this chapter proposes an efficient VNE algorithm for dynamic substrate networks with priori knowledge of temporal variations. The main idea is to subdivide the time horizon into time slots and to exploit the known information of the time dependent network connections in order to embed VNRs, not only for a single time slot, but also for multiple future time slots, minimizing, for each VNR, the migration cost between consecutive time slots. The "*migration cost*" is defined as the ratio between the number of consecutive time slots where the traffic has experienced a migration over the total planned time slots. Later, we provide a more detailed description for this metric.

This chapter is organized as follows. We initially provide the state-of-the-art for VNE in Section 3.2. Then, an insight of the motivations and contributions is presented in Section 3.3. Section 3.4 introduces the formulation of the problem and the network model, Section 3.5 presents the algorithm's steps and complexity and in Section 3.6 the relaxation procedure is described. Sections 3.7 and 3.8 present the performance evaluation from simulations and practical implementation, respectively. Finally, Section 3.9 concludes the chapter.

3.2 Related works

In this subsection, we firstly consider the main trends in the literature for VNE, and, secondly, we discuss the satcom-based VNE solutions.

3.2.1 Main VNE trends in literature

The relevant literature comprises a large variety of different VNE solutions. In fact, the general NP-hardness of VNE problems (44) has inspired different objectives and solving strategies. Some detailed and recent surveys can be found in (25; 45; 46). Online solutions for VNE are one of the main common trends among the proposed implementations (47; 48; 49; 50; 51; 52), where node and link mapping are computed either independently, which

most of the times provides an heuristic approach (53; 54; 55), or in a joint optimization (56; 57; 58; 59), which usually is an higher quality solution. In Section 3.3.1, we explain why online solutions are not the most efficient solution for satellite networks.

In the literature, some *dynamic* VNE solutions have been proposed. (60) considers a dynamic scenario but the dynamicity is provided by the time varying VNRs over a static substrate network. Indeed, the algorithm applies online re-mapping with the main objective of maximizing the acceptance ratio, given the actual demand, without giving priority to the migration cost. Authors in (61) investigate a time-dependent VNE algorithm where the VNR's migration cost is taken into consideration. However, in this case the migration process is driven by a reward system with the objective of improving the average network utilization and maximizing the acceptance ratio, without considering any migrations caused by a dynamic substrate network. Few other implementations are considering a dynamic substrate network. Authors in (62) and (63) propose resource mapping algorithms for satellite networks. However, these implementations are mostly relying on node caching optimization systems to maximize the network flow and acceptance ratio, without considering any migration cost.

3.2.2 Satcom-based VNE

Some VNE solutions specifically target satellite networks. Few satcom-based VNE implementations with a dynamic substrate graph can be found in the literature, in addition to the ones presented above. (54) and (64) address this problem. In both cases, the topology of the network changes over time. However, the algorithms are proposed as online algorithms, which means that, at each instant, the algorithm has to verify the entire network topology. Those VNRs, affected by the link change, will be re-mapped. This approach, as the previously mentioned online solutions, might produce a considerable high number of migrations over time per each VNR. While for terrestrial infrastructure, migrations might not be a relevant issue, for SatCom nodes, an excessive amount of migrations might significantly affect the network performance. In fact, migrations over satellite networks are not often smooth because of some temporal loss of service or dropped sessions. In addition, migrations require considerable signalling which creates traffic overhead and might be prone to errors.

Last, but not least, the majority of the proposed VNE works lack of testing the algorithm in a built-in testbed. Clearly, this is the most interesting and challenging aspect of the imple-

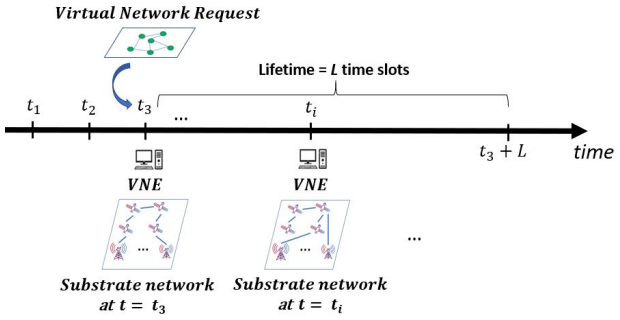


Figure 3.1: Traditional online VNE solution for dynamic substrate network

mentation, where the algorithm faces the technical effects of a software/hardware emulated system.

Given these premises, from theoretical perspective, a relevant attention should be given to the VNR migration cost, when the substrate network is highly dynamic, such as satellite networks. From the practical side, a testbed should be provided to validate the presented algorithm and to help understanding the pros and cons of its integration in a real network. To the best of our knowledge, a VNE algorithm which exploits the priori time-variation of NGSO constellation, in order to minimize the migrations of VNRs over their lifetime, has not been proposed before.

3.3 Motivations and Contributions

In this section, we firstly provide a comprehensive analysis about the relevance of an ad-hoc topology-aware VNE solution for satellite networks and the added-value of a built-in testbed to validate the results. Secondly, we detail our contributions.

3.3.1 Motivations

The literature review has highlighted how online VNE solutions are important and preferred for their simpler and efficient nature.

Figure 3.1 depicts a traditional online VNE approach for dynamic substrate networks (54). The time domain is divided into time slots such that the substrate network can be considered static within each time slot. A random VNR arrives at the time slot $t = t_3$, with a lifetime L. This means that the VNE algorithm is asked to guarantee the embedding, at least, until the time slot $t = t_3 + L$. At the time slot t_3, the VNE decisions have been

made based on the current status of the substrate network. However, at time slot t_i, the substrate network has changed, and it might happen that the VNR is affected by this change. In this case, the VNE has to be computed again, providing a different embedding and, consequently, triggering the traffic migration to the new path. This process is repeated until the VNR expires. Clearly, this approach lacks a global view in VNE optimization and may result in a typical suboptimal solution. A suboptimal solution might cause higher number of traffic migrations which corresponds, not only to economic expenses, but also to network performance degradation such as temporal outage, signalling, satellite handovers.

More precisely, from a network provider point of view, computational and/or technical costs might be involved for each VNR's migration. Clearly, the migration cost might be dependent on several factors such as the used technologies, the quantity of traffic to be migrated and, quite relevant, the considered layer of International Organization for Standardization/Open Systems Interconnection (ISO/OSI) protocol stack. For example, at the physical layer, migrations correspond to changes of hardware settings (antenna direction, etc.). At the layer 2, the time needed to re-configure one or several path might correspond to packet losses, frequent handovers, data plane signalling, which most likely has to be faced with some ad-hoc algorithms.

At the networking layer, the change of routing paths corresponds to modification of routing tables while, at the application layer, migrations are more related to the management of necessary computability resources to solve the complex VNE problem. In this chapter, we refer to the *migration cost* as the number of times, in percentage, that each VNR has to be migrated from one path to another one, due to topology variation, along its lifetime. It is worth underlining that this metric has a relevant impact on the computation time. In fact, on one side, the more planned time slots are considered, the lower the migration cost, in probability, due to a longer analyzed period of time. On the other side, the more the time slots, the higher the complexity of the problem, i.e. higher computation time. In this chapter we analyze this trade-off, driven by the choice of the number of time slots to be planned for.

In the VNE research field, there are widely used common metrics to evaluate the quality of VNE solutions such as *load balancing* (57; 60), *energy-saving* (65; 66), *acceptance ratio* (49; 51; 62; 63) and *computation time* (50; 67; 68). However, it is evident that the *migration cost* should be considered as a relevant metric to evaluate the VNE's performance and it should be given sufficient attention, especially in high dynamic scenarios such as the one

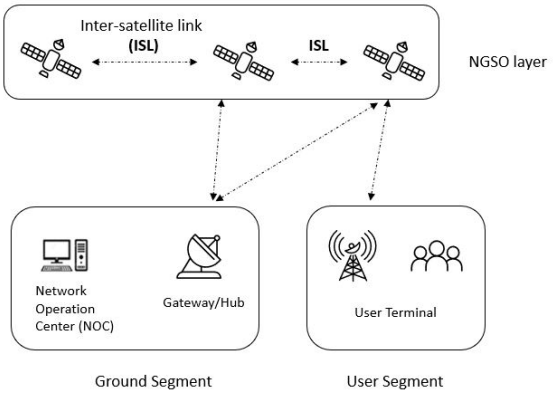


Figure 3.2: Migrations over terrestrial/NGSO network

considered in this chapter. Figure 3.2 depicts a global view of the considered scenario. A VNR, which is embedded over a satellite network, might be required to be migrated or forwarded toward a different satellite. This clearly does not happen often for GSO networks, known to be very stable and static, but quite regularly for NGSO.

The last relevant motivation for this work has a more practical meaning. The VNE literature lacks overall of practical validation of the VNE solutions. Usually, VNE algorithms are simulated and validated in Matlab. While this approach is time effective in comparing multiple algorithms and flexible in analyzing several performance, it considers ideal settings of the scenario. However, this approach might not highlight hidden details, coming from the real world. For high dynamic scenarios, for example, the traffic is likely to be migrated over time, and the effect of a migration on the experienced quality of service (e.g. packet lost) is difficult to be simulated in Matlab. This is the main reason why we developed and used a built-in testbed.

3.3.2 Contributions

Considering the highlighted trends of VNE's literature in Section 3.2 and the motivations for our work in Section 3.3.1, our contributions are summarized as the following:

- We propose a Dynamic Topology-Aware VNE (DTA-VNE) algorithm that minimizes the average migration cost for dynamic network requests over planned time slots. We employ a linearization process, to transform the original non-linear objective into a linear form;
- Since the DTA-VNE algorithm is based on a MBLP formulation, its complexity in-

creases exponentially with the network size and the planned time slots. To tackle this, we propose DTA-R, a relaxed version of DTA-VNE, which mitigates binary constraints and allows to study performance-complexity trade-off;

- We study the impact of the number of planned time slots on the trade-off between the migration cost and the computation time, comparing the results to the online shortest-path and load balancing algorithms. The proposed algorithms have proved to avoid unnecessary migrations already with few time slots considered, while keeping the computation time a reasonable value;

- As the literature lacks of VNE practical validations, we test the feasibility of our developed VNE algorithms via a MultI-layer awaRe SDN-based testbed for SAtellite-Terrestrial networks, MIRSAT, which employs the realistic topology of a MEO satellite system (69). The implementation results show that, while analyzing the traffic of a VNR, each migration foresees a considerable outage. MIRSAT intents to further motivate the prioritization of the migration cost in the optimization problem.

3.4 Network Model and Problem description

3.4.1 Network Model

We model the integrated satellite-terrestrial substrate network as a directed weighted graph $G_s = (N_s, E_s)$, where N_s is the set of substrate nodes and E_s is the set of substrate edges. Given u and v a pair of substrate nodes in N_s, we define $c(u, v)$ as the available capacity of the link (u, v) and $p(u, v)$ as the propagation delay. VNRs are randomly generated as end-to-end connection between random pairs of substrate nodes in the set N_s, given m the source node and d the destination node. In other words, the node mapping is already defined. While this might seem a strong assumption, in the following we provide two reasons to motivate this choice. Given the intent to validate the VNE algorithm in the built-in testbed, we visualize the VNR as client-server requests. This gives already defined nodes in the substrate network (source or client and destination or server). In addition, in order to have a validation consistent with the simulation results, we avoid to consider node related constraints (e.g. CPU) which is not trivial to be simulated in the testbed. Each VNR demands a minimum data rate b, a maximum tolerated latency l, it has a lifetime L time slots and it arrives at

time slot t_a. Therefore, each VNR's mapping is computed for the interval $[t_a, t_a + L]$ time slots. It is relevant to clarify that the unit of measure for the lifetime and simulation time is the "time slot". However, the formulation and results do not depend on the real duration of the time slots. Nevertheless, in Section 3.7, a real duration value is computed from the use-case and provided to give consistency and truthfulness to the emulated scenario. The VNRs follow a poisson distribution with average arrival rate λ VNRs per time slot. We assume that the topology of VNRs does not vary over time. In order to make the notations easier and more comprehensible, we do not assign the VNR's index to each VNR related parameter introduced before.

We consider the time-slotted system and assume the system operates for t_{sim} time slots. Despite the substrate network can be considered static within each time slot, the modelling variables should be considered as function of time. Indeed, we introduce the time dependency in the substrate graph notations. Accordingly, the substrate graph is modelled as a time expanded graph, following the approach proposed in (70; 71), $G_s^t = (N_s, E_s^t)$, with $c(u, v, t)$ as the available capacity of the substrate link (u, v) at the time slot t. It is worth noting that, while E_s^t and, consequently, the link resources, $c(u, v, t)$, are dependent on time, due to both the temporal topology variation and the consumption of resources over time assigned to the incoming VNRs, the set N_s, on the opposite, is not dependent on time because we assume that the available substrate nodes do not change over time. The same applies to the propagation delay.

3.4.2 Problem Formulation

In this section, we formulate the VNE embedding algorithm running for one VNR at a time. Considering that each VNR is simulated as an end-to-end request between two randomly selected substrate nodes, the node mapping is already defined. Thus, in the following, the VNE formulation is restricted to only the link mapping. Our objective is to minimize the migration cost, taking into account the known temporal description of the substrate network topology. Before presenting the formulation of the objective function, we initially introduce the following flow variable:

$$z_{uv}^t = \begin{cases} 1, & \text{if the VNR is mapped in link } (u, v) \text{ at time slot } t, \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

The algorithm focuses on the difference of link mappings between consecutive time slots, for each substrate link (u, v). In other words, the aim is to keep, as much as possible, a consistent link mapping over multiple consecutive time slots in order to minimize the migration of the VNR's traffic due to topology variations. Thus, given that the algorithm works with differences, the absolute value is required to avoid negative values, while comparing the link mapping between consecutive time slots (3.2).

$$|z_{uv}^{t+1} - z_{uv}^{t}|, \quad \forall (u,v) \in E_s^t, \tag{3.2}$$

At this stage, we introduce the following assumption. We do not give importance to how many links have changed between consecutive time slots, but, instead, to the presence or absence of any change. In other words, while comparing the mappings between consecutive time slots, if one link changes or all links are different, it is considered equally as a migration, i.e. binary migration variable, introduced in the following, equal to 1. It is worth underlining that this does not force the objective function to be always null. In fact, the objective function keeps minimizing the percentage of different consecutive mappings and the results show that, almost always, migrations occur if VNRs between at least one non-terrestrial node are involved. Accordingly, the algorithm is based on the following function:

$$f(t) = \begin{cases} 1, & \text{if } \sum_{(u,v)\in E_s^t} |z_{uv}^{t+1} - z_{uv}^{t}| > 0, \quad \forall t \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

$f(t)$ is a binary indicator function, which has the value "1" if there is at least one difference between the link mapping at time slot t and the one at time slot $t+1$. Based on the previous definition, equation (3.4) shows the initial formulation of the objective.

$$\min_{z_{uv}^t} \left(\frac{\sum_{t\in[t_a, t_a+T]} f(t)}{T} \right), \tag{3.4}$$

The objective function (3.4) minimizes the migration cost, previously defined as the percentage of migrations experienced by the VNR, between consecutive time slots, over T planned time slots.

It can be noticed that the function is not linear. In fact, in Section 3.5, we provide an explanation of the linearization process and the obtained final linear objective function.

In the following, the constraints of the problem are presented.

$$z_{uv}^t \cdot b \leq c(u,v,t), \quad \forall (u,v) \in E_s^t, \forall t \in [t_a, t_a + T], \tag{3.5}$$

$$\sum_{w \in N_s} z_{mw}^t - \sum_{w \in N_s} z_{wm}^t = 1, \quad \forall t \in [t_a, t_a + T], \tag{3.6}$$

$$\sum_{w \in N_s} z_{dw}^t - \sum_{w \in N_s} z_{wd}^t = -1, \quad \forall t \in [t_a, t_a + T], \tag{3.7}$$

$$\sum_{v \in N_s} z_{vu}^t - \sum_{v \in N_s} z_{uv}^t = 0, \quad \forall t \in [t_a, t_a + T] \quad \forall u \in N_s \setminus \{m, d\}, \tag{3.8}$$

$$z_{uv}^t \in \{0, 1\}, \quad \forall (u,v) \in E_s^t, \tag{3.9}$$

$$\sum_{u \in E_s^t} z_{uw}^t + \sum_{u \in E_s^t} z_{wu}^t \leq 2, \quad \forall t \in [t_a, t_a + T], \quad \forall w \in N_s, \tag{3.10}$$

$$\sum_{(u,v) \in E_s^t} z_{uv}^t \cdot p(u,v) \leq l, \quad \forall t \in [t_a, t_a + T], \tag{3.11}$$

$$\sum_{(u,v) \in E_s^t} z_{uv}^t \leq h, \quad \forall t \in [t_a, t_a + T], \tag{3.12}$$

Constraint (3.5) states that, for each time slot and for each substrate link (u, v), the assigned data rate cannot exceed the actual available capacity. It is relevant to clarify the reason why the minimum data rate is being upper bounded by the link capacity. Considering that the proposed algorithm is formulated for an SDN-enabled scenario, the SDN controller (responsible for executing the VNE algorithm) initially assigns the minimum data rate to be guaranteed for each VNR (which cannot clearly be embedded in links with insufficient capacity). Given the ability of the SDN controller to constantly check for the traffic usage of any mapped VNR, a higher data rate can be assigned real-time if any VNR requires that (higher data rate might happen e.g. unexpected peak of traffic), only in case of available capacity. Constraints (3.6), (3.7) and (3.8) refer to the flow's conservation law. Indeed for (3.6) and (3.7), given the source and destination substrate nodes of the VNR, the sum

of outgoing flows from the source node m has to fully flow over the network because the demanded data rate should be guaranteed. The same applies for the incoming traffic in the destination node d, with a negative sign. (3.8) guarantees the flow's conservation law for the intermediate nodes, where the sum of incoming flows has to always match the sum of outgoing flows.

Equation (3.9) defines the integrity constraint for the flow variable. From a physical point of view, the integer flow variable means that splitting-path is not accepted in this configuration, so either the flow is passing through a substrate link, or it is not.

The constraint (3.10) guarantees the absence of loops. This is relevant because the objective function (3.4) minimizes the link mapping differences between consecutive time slots, but it does not guarantee that loops are avoided. In addition, constraints (3.6)-(3.8) guarantee the flow's variable conservation, but they also do not guarantee that loops are avoided because loops would still keep, the flow's conservation law, valid. This is the reason why it is important to set the constraint (3.10) where, for each substrate node, the sum of incoming flows and the outgoing flows, cannot exceed "2". This value is given by the fact that each substrate node can, at most, manage two flows per VNR (if it is an intermediate node), one incoming and one outgoing, giving "2" as total sum. If the sum of flows is greater than "2", it means that there is a loop. Constraint (3.11) keeps the end-to-end propagation delay, computed as the sum of the delay of each link from source to destination, lower or equal to the tolerated latency. It is important to compare the propagation delay to the maximum latency allowed by the VNR, especially in satcom, where delays may not be negligible, to avoid that low-latency communications are affected by large delays, which makes the scenario not realistic anymore. Finally, constraint (3.12) reduces the solution's space, limiting the search space to the paths of length h. In Section 3.5, we highlight the impact of the value h on the complexity of the problem. Intuitively, the value h drives the trade-off between the quality of the solution and the computation time. Very low values of h will limit the solution's quality while keeping the computation time low. On the opposite, higher values of h will improve the solution at the expense of the computation time. The value h is very much dependent on the number of links and possible paths that are available over the network.

3.5 The Dynamic Topology-aware VNE algorithm

The formulation presented in the previous section is not linear. In this section, we provide a comprehensive description of the linearization process, the overall procedure of the proposed Dynamic Topology-aware VNE (DTA-VNE) algorithm and the problem's complexity.

3.5.1 Linearization process

The functions (3.3)-(3.4) are not linear due to the presence of the absolute value. In order to linearize them, we use the approach presented in (72). For the sake of simplicity, we describe the process for one generic link (u, v) at the generic instant t. Thus, we can consider the initial objective function, without summations, as:

$$\min(|z_{uv}^{t+1} - z_{uv}^{t}|). \tag{3.13}$$

We define a second variable x_{uv}^{t} and add the two following constraints:

$$(z_{uv}^{t+1} - z_{uv}^{t}) \leq x_{uv}^{t},\ \forall (u, v) \in E_s^t, \tag{3.14}$$

$$-(z_{uv}^{t+1} - z_{uv}^{t}) \leq x_{uv}^{t},\ \forall (u, v) \in E_s^t, \tag{3.15}$$

Finally, the linear objective function is written with the new variable as:

$$\min(x_{uv}^{t}). \tag{3.16}$$

The variable x_{uv}^{t} is introduced to substitute the absolute value in (3.13). However, as previously mentioned in Section 3.4.2, we are interested to use a migration variable in the objective function which denotes the presence or absence of migrations between consecutive time slots. Since the variable x_{uv}^{t} counts the differences between the flow variables between consecutive time slots, for each substrate link, separately, we introduce, according to (3.17), the migration variable y^t, which does not count how many links differ between consecutive time slots, but, rather, only denotes the differences. This is the reason why it is an integer variable:

$$y^t = \begin{cases} 1, & if \sum_{(u,v)\in E_s^t} x_{uv}^t > 0, \\ 0, & \text{otherwise} \end{cases} \tag{3.17}$$

Table 3.1: Formulation Variables

Variable	Description
z_{uv}^t	binary flow variable to indicate if the VNR is embedded in (u, v) at the time slot t
x_{uv}^t	absolute value $\lvert z_{uv}^{t+1} - z_{uv}^t \rvert$ defined for each substrate link (u, v)
y^t	binary variable to indicate if there is a migration from time slot t to $t+1$

Finally, the original problem is reformulated as follows:

$$\begin{aligned} &\min_{y^t, z_{uv}^t} \left(\frac{\sum_{t\in[t_a, t_a+T]} y^t}{T}\right), \\ &\text{s.t. } (3.5) - (3.12), (3.14), (3.15) \end{aligned} \tag{3.18}$$

Table 3.1 summarizes the optimization variables.

3.5.2 DTA-VNE algorithm

DTA-VNE solves the link mapping for each VNR, separately and independently, in a sequential way. As highlighted in Section 3.3, the core idea is to plan, in advance, the VNR embedding for multiple time slots. Thus, we firstly focus the attention to the single VNR mapping (Algorithm 1). Secondly, we present the main environment (Algorithm 2) of the simulation, which manages the input and output of every instance of Algorithm 1, the VNRs generation and the resources update over time.

Algorithm 1 takes, as input, the current status of the substrate network (topology and resources), the demanded data rate and the end nodes of the VNR, and the time slots to be planned. Thus, the embedding is computed for the current time slot and for T following time slots (results will be shown for different values of the parameter T). Lines 6-7 are solved and, if a feasible solution exists, the embedding for T time slots is provided as output.

In the following, the pseudo-code for Algorithm 2 is presented and discussed. The simulation runs for t_{sim} time slots (main *for* cycle) where each iteration lasts one time slot. For each iteration, the priority is given to the VNRs, embedded in the previous time slot, which

Algorithm 1: Single VNR mapping for the interval $[t_a, t_a + T]$

```
1  Input
2  current substrate network status G_s^t, ∀t ∈ [t_a, t_a + T];
3  demanded data rate b, tolerated latency l;
4  m and d, source and destination substrate nodes;
5  T time slots to be planned;
6  Solve (3.18)
7     s.t. (3.5)-(3.12), (3.14), (3.15)
8  if successful then
9  |   Output
10 |   link mapping for T time slots;
11 end
```

have to be re-embedded, solved by Algorithm 1 via the well-established LP solvers, in case the link mapping they have been assigned with, is not available anymore due to network topology variation.

After this step, once the queue of VNRs to be re-mapped is empty, in case some have expired, the system releases the corresponding substrate resources. Afterwards, the new arrived VNR(s), in case any, will be mapped via Algorithm 1. After each VNR mapping, the substrate available capacities are updated.

3.5.3 Complexity's Analysis of DTA-VNE Algorithm

To analyze the complexity of DTA-VNE, it is enough to consider Algorithm 1, given that Algorithm 2 is only managing different instances of the first one. Algorithm 1 (lines 6-7) is solved through an exhaustive search which will be interrupted by the first solution that makes the objective function null, i.e. no migrations over T time slots. Even though there is an exhaustive search, the number of feasible paths between two given nodes (source and destination of the VNR) is strongly dependent on the topology of the substrate network. Given that the considered substrate graph is significantly simpler than a complete graph, this considerably reduces the complexity of the problem. We assume g the maximum degree of the substrate graph, such that $g < |N_s| - 1$, with $|N_s|$ the number of the substrate nodes. Thus, the complexity of finding all feasible paths of length h, given the two end nodes, is upper bounded by $O(g \cdot (g-1)^{h-2})$. This formula is obtained by the fact that, starting from the source node, there are at most g possible adjacent nodes. Then, from the second until the penultimate node, there are at most $g - 1$ possible adjacent nodes that can be chosen as

Algorithm 2: Dynamic Topology-aware VNE

```
Initialization
Substrate network definition G_s^t, ∀t ∈ [t_0, t_sim];
VNR parameters (demanded physical resources, arrival distribution, end-to-end
  nodes);
for t ∈ [t_0, t_sim] do
    for VNR(s) to be migrated do
        execute Algorithm 1;
        if successful then
            update residual capacities of substrate links;
        end
    end
    for expired VNR(s) do
        release assigned substrate resources;
    end
    for new VNR(s) arrived do
        execute Algorithm 1;
        if successful then
            update residual capacities of substrate links;
        end
    end
end
```

next hop, given that the flow does not go back from the link it has arrived. Considering h the path length, there are $h+1$ nodes in total over the path, among which 2 nodes are source and destination. The second node of the path is considered in the first selection which has g possibilities. Finally, there will be $h-2$ nodes to be selected in the path among at most $g-1$ nodes every time. It is worth noting that the provided complexity is an upper bound because the value g is the maximum degree of the substrate graph. In addition, given that the objective function considers the combination of the single time slot embedding over T time slots, the total complexity is upper bounded by $O((g \cdot (g-1)^{(h-2)})^T) \sim O(g^{T \cdot (h-1)})$ if $g >> 1$. This highlights the impact of T, h and the density of the substrate graph (g), on the complexity.

3.6 LP relaxation

While the non-linearity of (3.13) is solved by the above-mentioned linearization process, the binary nature of the flow variable z_{uv}^t exponentially increases the computation time with the increase of the planned time slots and/or the network size. As the computation time

is crucially relevant for online VNE algorithms and considering the size and dynamicity of the targeted substrate networks, i.e. MEO and LEO constellation, we propose a relaxed version of DTA-VNE, named DTA-R. To obtain DTA-R, we relax the binary flow variable, in constraint (3.9), to continuous values. We underline that the exponential complexity of DTA-VNE is given by the binary constraint on z_{uv}^t, because the migration variable y^t grows only linearly with T, which is not large in practice. For this reason, the relaxation process is applied only to z_{uv}^t.

In addition, we introduce the following penalty function

$$P(\bar{\boldsymbol{z}}) = \sum_{(u,v)\in E_s^t} z_{uv}^t - {z_{uv}^t}^2 \tag{3.19}$$

to penalize the values of the flow variable, far from the integer extremes 0 and 1. We run an iterative process which stops when $|q^{i-1} - q^i| < \varepsilon$, with q^i be the solution at the i-th iteration and ε a very small number. Since (3.19) is not convex, we introduce the first order approximation which results as

$$P(\bar{\boldsymbol{z}}) = \sum_{(u,v)\in E_s^t} z_{uv}^t + {\bar{z}_{uv}^t}^2 - 2z\bar{z}_{uv}^t, \tag{3.20}$$

with $\bar{z}_{uv}^t$ the solution at the previous iteration. With the addition of $P(\bar{\boldsymbol{z}})$ to the objective function (3.18), the final objective function of DTA-R is expressed as

$$\begin{aligned} &\min_{y^t, z_{uv}^t} \left(\frac{\sum_{t\in[t_a, t_a+T]} y^t}{T} + \gamma P(\bar{\boldsymbol{z}})\right), \\ &\text{s.t. } (3.5) - (3.8), (3.10) - (3.12), (3.14), (3.15) \end{aligned} \tag{3.21}$$

with $\gamma > 0$ as the penalty weight parameter.

Once the iterative process terminates, the rounding technique follows a deterministic approach, where feasible, as proposed in D-ViNE (57). To understand the performance losses, we provide the trade-off analysis between DTA-VNE and DTA-R, in terms of average migrations and computing time for few planned time slots.

Figure 3.3 shows the loss in terms of average migrations, for multiple values of planned time slots. It can be noticed that when T = 2, there is almost no difference and, anyway, the algorithm does not perform well in both cases because there is almost no planning. When

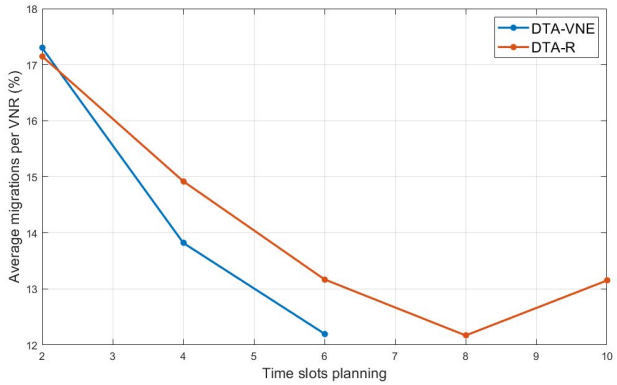


Figure 3.3: Comparison of average migration cost vs time slot planning between integer formulation (DTA-VNE) and relaxed one (DTA-R)

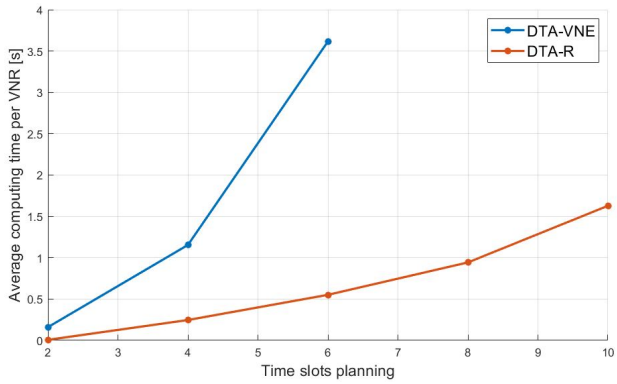


Figure 3.4: Comparison of average computing time vs time slot planning between integer formulation (DTA-VNE) and relaxed one (DTA-R)

T increases, a difference can be noted. Despite that, the relaxed version allows to go further with the planning, such as T = 8. This evaluation aims to show that the relaxed version experiences a reasonable loss in performance and it makes sense to use it for performance evaluation, even for MEO constellations.

The other side of the trade-off is depicted in Figure 3.4, where the gain in computing time is shown. DTA-R shows that even T = 10 has still a reasonable complexity to be computed.

3.7 Performance Evaluation with Simulations

In this section, we firstly describe the simulation setup and, secondly, we provide several performance evaluation analysis, while comparing DTA-R to online load balancing and shortest-path baseline algorithms.

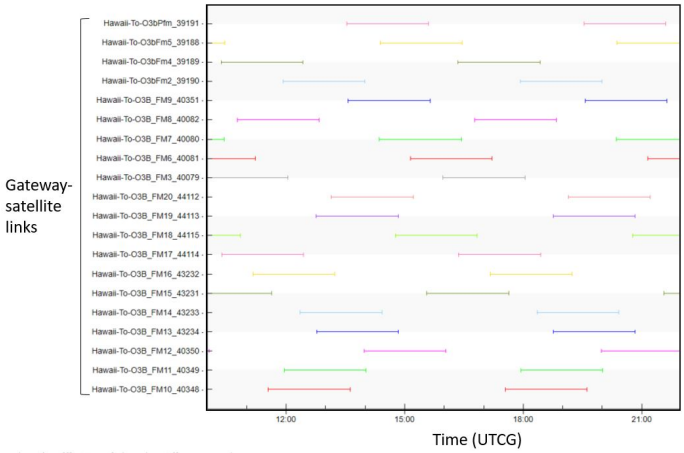


Figure 3.5: Access time scheme of Hawaii gateway to the 20 O3b satellites over 12 hours

3.7.1 Simulation setup

The substrate network is an integrated satellite-terrestrial network. We considered the SES O3b-MEO constellation with real location data for the MEO gateways (69) across the globe. The constellation is composed of 20 MEO satellites while the ground segment is composed of 9 terrestrial gateways. We additionally introduce Inter-Satellite Links (ISLs) among the MEO satellites, as a circular shaped topology. Thanks to the Systems Tool Kit (STK) (73) software, we have obtained the time series of the satellite-gateway link for each satellite and for each gateway over the entire duration of the simulation. The orbit of MEO satellites has an altitude of 8063 km, and considering that, for the purpose of this algorithm, we have to subdivide the simulation into time slots such that connections are static within each of them, the duration of a single time slot is $\sim$ 15 minutes, when applied to the real time scale. Later on in the section, we show in details how this value is obtained.

In STK tool, once the location of the gateway and the constellation parameters are set, we can obtain, for each terrestrial gateway, the description of the access over a defined amount of time, for all satellites. In this case, Figure 3.5 shows the time-dependent connectivity of the terrestrial gateway, located in Hawaii, with the entire constellation.

The graph is taken from 12 hours of observation. From Figure 3.5, considering the time on the x-axis, it can be noticed that for approximately 15 minutes, the network remains static. Therefore, we assume the duration of each time slot equal to 15 minutes. The obtained value is the result of a trade-off between the number of time slots and the precision of their duration with respect to the real network behavior. In fact, from Figure 3.5, it can be deducted that,

in reality, the network is not static for exactly 15 minutes. A rounding to the closest multiple of 15 has been applied. For example, if the connection stops at 15.55, it will be counted as 16.00. While if it stops at 15.50, it will be counted as 15.45. While this might seem introducing an error $\in [0, 5-7]$ minutes, it should also be noticed the following. Each gw-sat connection lasts for 6 time slots, which correspond to 2 hours, 7200 seconds. In the reality, each sat-gw connection lasts for $\in [7023, 7029]$ seconds, therefore a maximum error of 177 seconds is introduced. This is due to the fact that whenever there is a rounding in one time slots, this is compensated from the rounding of the following slot. In other words, if the connection is between 15.55-16.10, this is reported as 16.00-16.15, which still keeps the 15 minutes duration.

In total, the simulation lasts $t_{sim} = 72$ time slots, which correspond to 18 hours of real time. As also highlighted above, the duration of the time slot is extremely dependent on the substrate network. However, the proposed VNE algorithms are not dependent on the real duration of the time slot and, in addition, a time slotting strategy with variable time slot duration would still be applicable. Figure 3.5 depicts the time-dependent connection for only one terrestrial gateway and it can be obtained for every terrestrial gateway. It is relevant to underline that, if we project the simulated time slots to a real case, intuitively, as we have just shown, the duration in minutes is dependent on the considered satellite network. This real duration of the time slots will be relevant later on in this chapter for the simulation results and the final discussion due to the following reason. The proposed algorithm is thought to be applied to a variety of dynamic networks such as LEO and MEO, which differ especially for the amount of time the status of the satellite network can be considered static. In fact, if we consider a MEO constellation, the link connectivity between a terrestrial gateway and a MEO satellite can remain up to 2 hours. When the orbit of the constellation is considerably lower (LEO), the duration of the time slot decreases significantly, even up to few minutes, depending on the SatCom related parameters such as the orbit's altitude.

From the terrestrial connections point of view, we consider that the gateways are also connected via terrestrial links (no variations over time) among themselves as a mesh fully-connected network. The available capacity of both terrestrial and satellite links is initially set equal to 300 Mbps. The source and destination nodes of each VNR are randomly selected among the substrate nodes to consider the worst possible scenario, i.e. when a priori prediction is not available.

The two following different cases for the source-destination random generation are being considered:

- **Case 1**: the source is randomly selected among the satellites and the destination is randomly selected among the terrestrial gateways;
- **Case 2**: source and destination are randomly selected among the entire set of substrate nodes.

The two use cases are selected to analyze the efficiency of the proposed algorithm, varying the diversity among the type of VNRs. In particular, use case 1 selects randomly the source and destination nodes which belong to two different segments, one satellite and one terrestrial. In this scenario, migrations are more likely to happen. In use case 2, instead, nodes belonging to the same layer might also be selected, thus, migrations might not be involved by default and the efficiency of this approach would reduce with respect to a greedy approach. For the sections 3.7.2 and 3.7.3, a comparison between these use cases is presented while, for the results in section 3.7.4, only use case 1 is considered since it was proved to be the most interesting scenario for applying DTA-R algorithm. The demanded data rate by each VNR is randomly generated in the interval [40,100] Mbps while two different values of latency requirement are randomly selected among [30, 1000] ms. The propagation delay $p(u, v)$ is set to 27 ms for the ground-sat links, 1 ms for ground-ground links and 3 ms for sat-sat links. The parameter h is fixed to 10. We compare DTA-R to the following online baseline algorithms:

- **Ref. 1: Shortest path** (64);
- **Ref. 2: Load Balancing** (54).

Both algorithms are also considering a node mapping optimization which is not taken into consideration in this chapter, thus they have been adopted to only link mapping optimization.

3.7.2 Average migration cost per VNR

In this first simulation result, the metric is the average percentage of migrations that each VNR experiences, during its lifetime. The results are shown for both cases, mentioned in the previous section. From Figure 3.6, several considerations can be done.

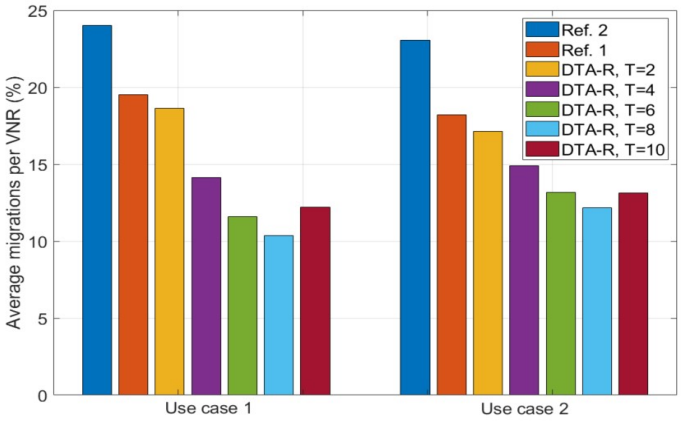


Figure 3.6: Average migration cost per VNR for use cases 1 and 2

First of all, it is important to remind that the metric *average migration cost* is expressed in percentage because, as stated previously, we refer to the migration cost as the ratio between the number of migrations, experienced by each VNR, over the lifetime. For example, if we analyze one single VNR and we assume its lifetime L time slots, it means that there are L transitions over time where the traffic might have to be migrated due to topology variation. If we assume that n migrations have occurred, the migration cost would be n/L. Then, at the end of the simulation, we take the average migration cost among all VNRs.

The proposed algorithm is compared to the baselines presented in 3.7.1. As mentioned above, the DTA-R is based on a planning of a defined number of time slots T. Accordingly, in the results, different values of T are analyzed, 2, 4, 6, 8 and 10. Figure 3.6 shows, in both use cases, that the average migration cost per VNR obtained with DTA-R is similar or considerably lower than the baselines' one, for all values of T. In addition, the more the value of T, the more DTA-R is efficient because the path with less migrations, over a longer period of time, is selected.

Furthermore, a comparison between the two use cases can be highlighted. Figure 3.6 depicts, on the left side, the use case 1 where migrations are more likely to happen while, on the right side, use case 2, where VNRs can be generated even among terrestrial nodes. It can be noticed that the benefit of DTA-R for use case 1 is slightly higher, with respect to the baselines, than in use case 2, with $T > 2$. For example, when $T = 4$, for use case 2, DTA-R gains $\sim$ 36 % with respect to baseline 2, while, on the opposite, in use case 1, DTA-R gains $\sim$ 41.2 %. However, even in the latter case, an advantage of DTA-R can still be appreciated, but the benefit is slightly lower than use case 1, due to the less dynamicity of the scenario.

It has also been noticed that for values higher than T = 8, the rounding procedures is more likely to fail, falling into a shortest path approach, which worsens the performance on average. While this might seem a limitation, it is worth underlining the following. Despite the type of algorithm or substrate network, when dealing with dynamic scenarios, such as NGSO, there should be an upper-bound in the number of priori planned time slots. In fact, it makes sense to plan in advance for some time, exploiting the priori knowledge, but it does not make sense to plan too much in advance, because the more distant the future, the more difficult to predict the status of the network. For example, in a MEO constellation, considering 6 or 8 planned time slots, it corresponds to a planning of already $\sim$ 2 hours.

3.7.3 Computation time

The second performance evaluation considers the computation time, which is a very critical analysis in this field. As depicted in Section 3.5.3, the complexity grows exponentially with the number of planned time slots T, therefore, the computation time is expected to be very sensitive to the increase of T. As shown in Figure 3.4, the DTA-R considerably reduces the computing time, and it allows to consider values of $T = 6$ or 8, without introducing long delays. Despite the type of DTA utilized, we should not expect lower values than baselines' ones, but, rather, the purpose of this comparison is to give an idea on which could be the ideal value of T, while keeping the solution time reasonable.

Table 3.2: Average computation time per single VNR embedding

Algorithm	**Case 1**	**Case 2**
Ref. 1	0.0056 s.	0.0013 s.
Ref. 2	0.0201 s.	0.0148 s.
DTA-R, T=2	0.0026 s.	0.0069 s.
DTA-R, T=4	0.26 s.	0.24 s.
DTA-R, T=6	0.55 s.	0.44 s.
DTA-R, T=8	0.94 s.	0.71 s.
DTA-R, T=10	1.74 s.	1.62 s.

Table 3.2 shows that, in both use cases, the computation time grows exponentially with T, as expected. For use case 2, the reason why there are overall lower values, is related to the fact that the computation time is an average over the simulation time. For the VNRs generated among terrestrial nodes, the computation time is almost none, since it is easier to find paths with no migrations, which reduces the overall average. This situation does not

happen in use case 1.

The evaluation on which is the right value of T, significantly depends on the scenario. For example, for delay-tolerant applications, such as multimedia content download, even 2 seconds per computation might still be a reasonable value. On the opposite, for those less delay-tolerant applications, this delay might not be reasonable anymore. Thus, the selection of the right value of T is very much use-case dependent.

It is worth underlining that the computation time metric has a relevant importance for another reason. If we consider that the algorithm takes, on average, the computation time depicted in the Table 3.2, this means that if there is a queue of tens of VNRs, waiting to be embedded, the last VNR(s), might experience a long delay because the algorithm is embedding in a sequential way. Obviously, VNRs might be re-arranged in the buffer following criteria such as priority, maximum allowed latency, etc. but it is still very important to control the computation time, considering the number of VNRs to be embedded. For the following performance evaluations, $T = 8$ is considered since, overall, it seems to be the most reasonable value.

3.7.4 Average network load over time

In this section, we evaluate the average substrate link load over time. This is a common metric, also known as load balancing, used to evaluate how the traffic is spread in the network. Indeed, the average substrate link load metric takes the capacity utilization of every substrate link (percentage) and computes the average over all links. For the comparison, the load balancing algorithm is considered as benchmark because it provides a very good solution.

It is important to underline that this metric is quite relevant in current transport networks. A low average value of utilization is equivalent to say that the traffic is fairly spread which also means that unexpected peaks of traffic might be more easily managed in the network, with lower probability of bottlenecks. In this context, the performance evaluation of this section aims to prove that the proposed approach provides reasonable results even in terms of load balancing. For the comparison, we fix some parameters for DTA-R such as the planned time slots $T = 8$.

Obviously, considering the average substrate link utilization as the metric, worse results of the DTA-R algorithm, compared to load balancing, are expected for any level of planned time slots. However, the promising result, which can be noticed in Figure 3.7, is that the

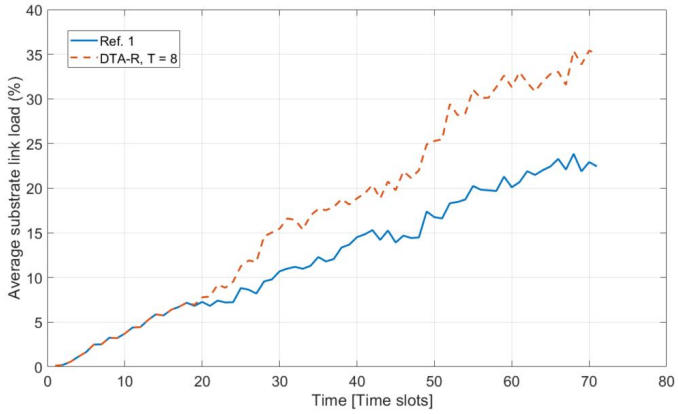


Figure 3.7: Average substrate load vs time

DTA-R algorithm provides values that follow the same trend of the load balancing ones, keeping the distance, with the load balancing graph, always a reasonable value. The more the time passes, the more the accommodated VNRs, and the higher the probability that DTA-R deviates from the better solution (Load Balancing). This is expected because indeed, the objective of DTA-R is not to reduce the average substrate load. In addition to that, the VNR is an end-to-end connection, which gives less degree of freedom to the embedding algorithm, if compared to cases where VNRs are composed by several links and nodes. If we consider for example, larger size networks, such as a LEO constellation instead of MEO, the story might be different and the difference might be larger. In this context, a joint (load balancing and migrations minimization) objective function could be considered, where complementary weights are given to the two objectives and different combinations of the weights can be studied in order to keep also the load, over the network, under control.

3.7.5 Average path length per VNR over lifetime

In this section, we analyze the average path length for the baselines and the DTA-R. As for the analysis presented in 3.7.4, this performance evaluation does not foresee to improve the results obtained by the Ref. 1 algorithm, which is a pure shortest path approach, but rather to show that the solutions of DTA-R do not explode, not only in average load, but also in terms of path length.

In fact, Table 3.3 shows that the average path length for every considered T value, is always not very far from the lowest one, i.e. Ref. 1. Despite this, a trend can be highlighted. The more the planned time slots, the higher the average path length. The reason for this

Table 3.3: Average path length (hops)

Algorithm	**Case 1**	**Case 2**
Ref. 1	2.76	3.09
Ref. 2	2.80	3.12
DTA-R, T=2	2.79	3.12
DTA-R, T=4	2.83	3.19
DTA-R, T=6	2.93	3.33
DTA-R, T=8	3.49	3.98
DTA-R, T=10	3.59	3.86

result is linked to the ability of the algorithm to find static links over T planned time slots. For intra-segment VNRs (from satellite to terrestrial node), it is more likely to have longer paths in order to minimize the difference between consecutive time slots rather than shortest paths but more dynamic over time. This is an additional reason why it is important to do not increase too much the number of planned time slots.

3.8 Experimental results in the built-in SDN-based testbed for VNE algorithm

In this section, we test DTA-R and the baselines algorithm into our testbed MIRSAT. The aim is to use MIRSAT to analyze the impact of the migrations on the real traffic, which cannot be easily simulated in Matlab. Initially we run Matlab simulations in an ideal case and we focus on the optimization aspect, showing the benefit of priori planning. However, simulations lack of precise measure of the traffic lost per each migration (which is not necessarily always the same). In fact, MIRSAT aims to give a precise description of the traffic lost per each migration. MIRSAT is an SDN-enabled testbed, with a centralized entity, the SDN controller, and a dynamic substrate network, i.e. the satellite network. SDN is a useful technology to test mapping algorithms, such as VNE, because it enables an easy translation of mapping decisions into forwarding rules, also known as flow rules. Deploying the routing control externally in the SDN controller, brings to a simplification of the network nodes, which forward the traffic according to the pre-computed path, instead of internally running routing algorithms. This relevant feature matches very well with the satellite context, where the majority of the network nodes are satellites, i.e. nodes with limited resources. Consecutively, satellite nodes are seen as Layer-2 enabled switches.

3.8.1 Experimental testbed setup

We developed MIRSAT (74) as an enhanced version of the testbed that we proposed in (75). In (75), the testbed has three main software components. The substrate network, emulated in Mininet (33), where the nodes are OpenFlow enabled switches, the RYU SDN controller (32) which has full control of the Mininet switches and the VNE algorithm in Matlab, directly connected to the SDN controller. MIRSAT is a more complete testbed, where we additionally propose the integration of the STK toolkit, connected to Mininet (Figure 3.8). This brings flexibility to the testbed, allowing to emulate, in Mininet, any satellite scenario that is simulated in STK. The testbed is fully running in a virtual machine with 10 GB of Random Access Memory (RAM), 1 Central Processing Unit (CPU) and 50 GB of storage capacity. In fact, once the scenario is defined in STK, the description of the substrate network topology and its temporal variations are translated into Mininet code.

The VNE algorithm retrieves the network topology at any time through the Representational State Transfer (REST) API of the SDN controller, and uses it as substrate network to embed any incoming VNR. Once the VNE problem is computed, the output is the link mapping of the arrived VNR, which is translated, by the SDN controller, into flow rules, and sent, through the OpenFlow channel, to the switches involved in the link mapping (source, destination and intermediate nodes). MIRSAT emulates the scenario described so far. This means that Mininet emulates the integrated MEO/terrestrial network, with properly emulated latency for MEO links. RYU controls each node (terrestrial and satellite) and, in Matlab, we run the three different algorithms, to test the performance. In order to make the emulation faster, we consider that each time slot lasts for 60 seconds, instead of 15 minutes (real value).

3.8.2 Migration cost

In this section, we analyze the traffic of a single VNR, over 8 time slots, which correspond to 7200 s. in the reality. In order to simplify the presentation of the results from the testbed, we reduce the duration of each time slot to only 60 s., instead of 15 minutes. This helps to have a good understanding of the transition time and the traffic lost when entering in a new time slot. In fact, if we kept the real duration, few seconds out of 15 minutes, would have not been clearly noted. In addition, the difference between DTA-R and baselines can be appreciated more easily. We use UDP traffic via *iperf* to test the connection between

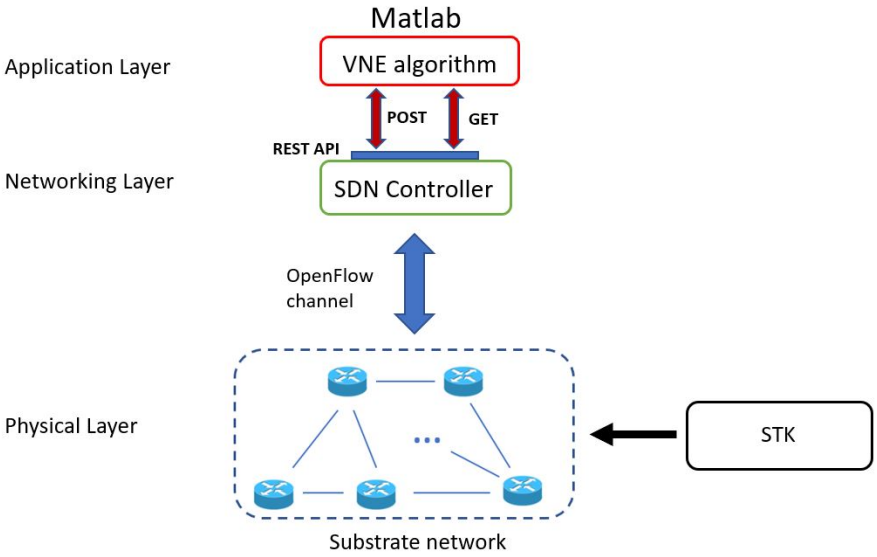


Figure 3.8: MIRSAT - ISO/OSI hierarchical description

source and destination nodes of a randomly selected VNR, among the arrived ones (Figure 3.9). As for theoretical simulations, the DTA-R with $T = 8$ is compared to Ref. 1 and Ref. 2 algorithms. Figure 3.9 shows the throughput behavior experienced by the VNR during the considered time, for the three different algorithms. To ease the comprehension, we have drawn a continuous line, external to the graph, with the description of the time slots. The drops of throughput to values close or equal to 0, correspond to an ongoing migration of traffic. In this scenario, it is evident that a period of outage between consecutive time slot, is experienced. This is due to the SDN controller, which takes few tens/hundreds of second to realize the new links. Once RYU can again see all the links in the topology, the traffic will flow towards the new paths. The outage period varies, depending on the amount of new links that the SDN controller has to discover. RYU can take up to $\sim$ 4-5 seconds to discover all new connections. The testbed results are consistent with the simulated ones. In fact, the average migration cost is considerably higher while considering the baseline algorithms. DTA-R proved to significantly reduce the migration cost.

In addition, we have run ping sessions between VNR's end nodes to retrieve the precise amount of lost packets and get a final percentage of outage. Table 3.4 provides the outputs from the ping sessions, where one ICMP packet is sent every second, thus, overall $\sim$ 450 packets are sent. The outage percentage achieved while using DTA-R has been reduced by $\sim 2.5 - 5\%$, compared to the baseline results.

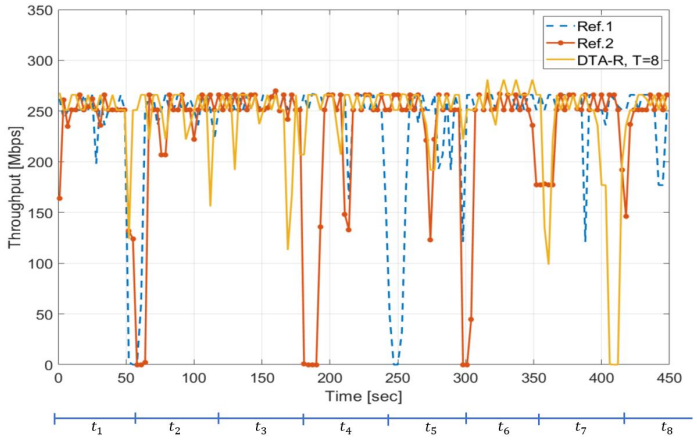


Figure 3.9: Comparison of throughput obtained via iperf session from one VNR

Table 3.4: Transmitted vs Received Packets

	Transmitted	**Received**	**loss (%)**
Ref. 1	451	430	4.6
Ref. 2	451	420	6.9
DTA-R, T=8	451	442	2.0

3.9 Summary

In this chapter, we proposed a formulation for a dynamic and topology-aware VNE algorithm with dynamic substrate network. The outcome of this work can be served as a guideline for performance evaluation of VNE on integrated satellite/terrestrial networks, which is useful to both satellite and terrestrial service providers. In fact, we have considered an integrated terrestrial/MEO satellite substrate network, with also MEO-MEO links. We simulated the real time series of O3b MEO constellation, via STK toolkit. Given a subdivision of the simulation into time slots, where the substrate network could be considered static within each time slot, the DTA-VNE algorithm was able to plan, in advance, the embedding of each VNR for a certain amount of time slots, with the objective of minimizing the migration cost over the planned time slots. Due to the exponential complexity of DTA-VNE with the increase of planned time slots and network size, we have proposed DTA-R, a relaxed version of DTA-VNE. Simulation results have shown that even DTA-R outperforms baseline solutions, such as shortest-path and load balancing algorithms, while considering the migration cost per VNR. We proved that the more planned time slots, the higher the gain of DTA-R. Simulation results have also shown the trade-off between the computation time and the number of planned time

slots. This is fundamental to determine the maximum allowed time slots, given the scenario (QoS requirements) and the size of the substrate network. In order to deeper investigate the application and impact of the proposed algorithm, we developed a proof-of-concept SDN-based testbed, MIRSAT, with a proper time-dependent emulation of the substrate network in Mininet. The testbed has shown the importance of avoiding migrations. In fact, we analyzed the traffic of one randomly selected VNR and we proved the outage of service, quantified as lost packets, experienced for each migration.

Chapter 4

Traffic-Aware Virtual Network Embedding with Joint Load Balancing and Datarate Assignment for SDN-Based Networks

4.1 Introduction

In the previous chapter, a study of link mapping for VNE in NGSO networks was discussed and a solution to minimize the handovers was proposed. Following the same trend, this chapter proposes a link mapping formulation for VNE. Traditionally, VNE is computed considering that the resource requirements are known. While this was a reasonable assumption when the optimization strategies were only at the initial stage, it became an obsolete approach, considering the dynamicity and load of the current information networks.

To fully optimize the embedding algorithms, the traffic from each embedded VNR should be periodically collected to, firstly, keep checking the actual current utilization of the assigned resources and, secondly, use a stochastic traffic description to optimize the assigned resources. From the infrastructure and implementation perspectives, promising technologies such as SDN (16) can support this method by collecting, in pre-defined intervals of time, the traffic statistics, and describing each VNR's traffic as a random variable with average and variance. In addition, real-time routing decisions and/or VNE adjustments can be executed by the

SDN controller based on the current network utilization. In this chapter, we propose a VNE algorithm for an SDN-based network, where an initial embedding is computed in the traditional way for each VNR for the first period of time. After the collection period, the traffic of each VNR is described as a random variable and a joint load-balancing and data rate assignment optimization is performed. The SDN controller supports the VNE algorithm by sending the traffic usage in an almost real-time manner.

4.1.1 Related Works

As the scope of this chapter is to investigate link mapping, in this section, some of the main link mapping algorithms, currently proposed in the literature, are analyzed. While mapping a VNR onto the substrate network, any proposed scheme has its own set of features/resources which drives the solution. The choice of the considered resources makes the proposed schemes different one each other. To facilitate the literature review, we consider a subset of features.

The first main trend of VNE was to consider the VNRs as static requirements, both in the network graph and/or the requirements. This was the first approach used when the priority was given to elaborate on different techniques to solve the VNE problem (57; 76; 77). Considering VNRs as static resource assignment is very limiting, especially in dynamic and heterogeneous scenarios such as 5G and satellite communications. In fact, several VNE implementations proposed a dynamic VNR description. The dynamicity has been introduced either in the substrate network (54; 64; 78; 79), in the VNR graph (60; 80) or in the online migration of some node and/or link mappings (61; 81; 82; 83; 84; 85). On one side, these works foresaw a real-time reconfigurability of the VNE, which make them a good support to several scenarios, from satcom to 5G well-known use cases, e.g. enhanced Mobile Broadband (eMBB), Ultra Reliable Low Latency Communications (URLLC) and massive Machine Type Communications (mMTC). However, they lack one of the main dynamic features of the current telecom networks, i.e., the time-varying behavior of the traffic. Authors in (86; 87) investigated this aspect. In these works, VNRs were simulated with time-varying requirements which might bring to reconfiguration of some embeddings over time. The periodic re-computations are proposed also in (88), where authors constantly re-compute the VNE for the allocated VNRs to increase the overall utilization of the substrate resources. Although the solutions therein are close to realistic scenarios, they performed the network dimensioning in a rather static way, where requirements of VNR(s) are fulfilled 100% all the time. In prac-

tice, this is normally not the case for modern networks (89). Some works introduced already the idea of directly considering the traffic of each VNR via the probabilistic description (90), which is one of the most realistic approaches to be used in VNE because it follows the nature of the traffic.

Despite that, two topics have not been yet addressed in the literature. The first one is related to the description of the traffic. Authors in (90) described the traffic with a probabilistic approach, however, each VNR's traffic was supposed to be known a priori, via the mean and variance. This assumption is not always realistic since, often, the traffic stochastic process is non-stationary and thus their statistical parameters evolve over time. The most realistic approach would be to observe the traffic for each VNR and extrapolate its mean and variance. The second relevant missing feature is related to the probabilistic approach, when embedding VNRs, introduced by the User Satisfaction Probability (USP). USP is defined throughout this contribution as the probability to satisfy the traffic demand. While, traditionally, VNRs are constantly assigned what they have required, in the reality, on the opposite, any network operator does not plan to provide maximum service 100% of the time to each user. Indeed, a relevant missing feature in VNE literature is the introduction of a guaranteed USP. This would increase the efficiency of the substrate network resources and, consequently, the acceptance ratio. Finally, VNR might be treated differently for their resource requirements or importance, and, consequently, be assigned with different USPs. Differentiating VNRs is not a new concept since some works already defined VNE with a priority scheme (91; 92; 93). However, these works focused only on the priority when sorting the VNRs to be embedded, without considering any optimization technique to be matched with different QoS that VNRs of different priority may have, e.g guaranteed USP, queuing delay. To the best of our knowledge, a link mapping algorithm for VNE which jointly optimizes the load-balancing and the assigned resources with a guaranteed USP, after describing the real-time traffic as a stochastic event, has not been presented before.

4.1.2 Motivations

As mentioned in section 4.1.1, the first main motivation of this work comes from the fact that traffic is never static or stationary in real-world scenarios. We aim to simulate, as realistically as possible, a scenario where several and different QoS VNRs send varying traffic over the network and to run the VNE algorithm in a realistic and dynamic context.

Secondly, as the infrastructure provider aims at optimizing its operation and profit, while guaranteeing the requirements of each VNR, real-time traffic monitoring is crucial. This approach requires a flexible network management, which can differentiate VNRs, monitor traffic from any mapped VNR and real-time change routing decisions based on internal policies, if needed. SDN is the promising candidate to provide the VNE with these features and improve the overall performances. In fact, SDN controller(s) can efficiently support VNE algorithms to manage the incoming traffic by describing the current utilization for each VNR and almost instantaneously (physical transmission time, between SDN controller and nodes, delays the process) translating the mapping decisions, coming from VNE, into forwarding rules, to let the traffic flow into the network.

In general, it is not sufficient to get a precise and realistic description of the traffic, for every VNR, but it is also important to differentiate VNRs, when providing them with the required resources, with a stochastic approach of USP. In fact, considering different priority VNRs, not only should be applied as the order which they are embedded in, as proposed in a few works in the literature, but also in the USP, which is likely to be different depending on whether the VNR has a high priority, e.g., emergency services, or lower priority, e.g., sensor networks, earth observations. This probabilistic approach reduces the assigned data rate, while keeping the USP under control. Some techniques to facilitate the embedding of high priority VNRs, especially in congested scenarios, such as the congestion ratio (65) are also introduced to increase the probability of having available capacity for higher priority VNRs. Finally, we specify that, while normally VNE includes both node and link mapping, in this contribution we prefer to focus on link mapping and consider the node mapping already computed. Different reasons drove this choice. In the literature there are several near-optimal solutions which compute node and link mapping in a coordinated manner. However, these solutions are time-consuming. In fact, simpler solutions, where node and link mapping are computed sequentially and independently, are typically proposed to speed up the computation process. As we intend to provide a solution computed in low time due to the traffic requirements, we followed the same approach. Consequently, since the focus of our algorithm is on traffic analysis and real-time link monitoring, it is convenient to prioritize the link mapping. We noticed that the analysis was complex and to simplify the readiness of this chapter, we decided to keep the node mapping known a priori.

4.1.3 Contributions

Considering the highlighted trends of VNE's literature in Sec. 4.1.1 and the motivations for this work in Sec. 4.1.2, our contribution is the following:

- As the VNE literature lacks of considering USP as one of the main KPI, we propose a SCA-VNE which minimizes the average link capacity utilization, given a minimum guaranteed USP. We consider different priority classes of the traffic and evaluate the performance in terms of acceptance ratio, average queuing delay and the USP, for each priority class.
- To tackle the exponential computation complexity of the formulated MBLP problem, we propose a relaxed version, SCA-R, which relaxes the binary constraint of the flow embedding variables with a penalty function to balance the convergence-performance trade-off. We show that SCA-R is a more scalable solution, with respect to SCA-VNE, since the time complexity of SCA-R grows significantly slower than the SCA-VNE's one, as the number of network nodes increases.
- Finally, the proposed VNE algorithm is validated via testbed experimental results. We build an emulated SDN testbed to collect traffic statistics from each VNR in real-time scenarios. Based on the current utilization of each VNR, SCA-VNE releases resources that are not utilized and assigns them to other incoming VNR. Both simulation and implementation results show that SCA-R outperforms existing solutions by reducing the acceptance ratio, in highly-loaded scenarios, and, by considerably minimizing the average queuing delay, especially for higher priority VNRs.

This chapter is organized as follows. Section 4.2 formulates the problem and Section 4.3.4 describes the SCA-VNE algorithm and the relaxed version SCA-R. Some initial simulations are presented in Section 4.4 and the performance evaluation, as well as the testbed description/results, are presented in Section 4.5. Finally, Section 4.6 concludes the chapter.

4.2 Network Model and Problem description

4.2.1 Network Model

We model the substrate network as a directed weighted graph $G_s = (N_s, E_s)$, where N_s is the set of substrate nodes and E_s is the set of substrate edges. Given u and v a pair of substrate nodes in N_s, we define $c(uv)$ as the available capacity of the link.

Every VNR is modelled as a directed graph: $G_v = (N_v, E_v)$, with N_v the set of virtual nodes and E_v the set of virtual edges. Each VNR is described via a fixed data rate b, a priority level p, a lifetime L time slots and a start time slot t_a. Given the defined priority level, a minimum USP P_g is guaranteed. The VNR will leave the network at $t_a + L$. The VNR generation follows a Poisson distribution (60) with average arrival rate λ VNRs per time slot. VNRs are randomly generated as end-to-end connections between any pair of substrate nodes in the set N_s, given m the source node and d the destination node. We assume that the topology of VNRs does not vary over the considered time frame. For each VNR, we define a counter t_{count} which keeps track of whether the VNR has registered a very low value of traffic and possibly the lifetime. For all VNRs, a common collection period $t_{collect}$ is defined. After the collection period, the traffic of each VNR is modelled as a random variable X, with mean value μ and variance σ^2 (we omit the VNR subscript index for ease of presentation). The simulation time is subdivided into time slots of duration t_{slot}. In addition, let us denote θ as the minimum required serving percentage of the VNR and t_{exp} as the number of time slots the VNR experiences traffic smaller than θ. Finally, we define R_h, R_m and R_l as the congestion ratios for the high, medium and low priority VNRs, respectively.

4.2.2 Problem Formulation

In the following, the objective and constraints of the problem are presented. The objective function (4.1) jointly optimizes the load-balancing, which aims at spreading the traffic in the substrate network as more as possible, while minimizing the assigned data rate. To simplify the reading, we summarized the variables and parameters in Table 4.1. The variables are in bold. Two problem variables are defined as summarized in Table 4.1.

The initial joint objective can be written as:

Table 4.1: Variables and parameters in eq. (4.1) - (4.9)

$\boldsymbol{z_{uv}}$	binary variable to indicate if the VNR is embedded in $uv \in E_s$
$\boldsymbol{x}$	data rate
E_S	set of substrate links
$c(uv)$	availability capacity on the link uv
R	congestion ratio
N_S	set of substrate nodes
P_g	minimum USP to be guaranteed
μ, σ^2	average and variance of simulated traffic

$$\min_{z_{uv},x} \frac{1}{|E_s|} \sum_{uv \in E_s} \frac{z_{uv} \cdot x}{c(uv) + \epsilon}, \tag{4.1}$$

with ϵ a very small number. Given the following constraints:

$$z_{uv} \cdot x \leq R \cdot c(uv), \quad \forall (uv) \in E_s, \tag{4.2}$$

$$\sum_{w \in N_s} z_{mw} - \sum_{w \in N_s} z_{wm} = 1, \tag{4.3}$$

$$\sum_{w \in N_s} z_{dw} - \sum_{w \in N_s} z_{wd} = -1, \tag{4.4}$$

$$\sum_{v \in N_s} z_{vu} - \sum_{v \in N_s} z_{uv} = 0, \quad \forall u \in N_s \setminus \{m, d\}, \tag{4.5}$$

$$z_{uv} \in \{0, 1\}, \quad \forall (uv) \in E_s, \tag{4.6}$$

$$z_{uv} + z_{vu} \leq 1, \quad \forall (uv) \in E_s, \tag{4.7}$$

$$\frac{1}{2}[1 + erf(\frac{x - \mu}{\sigma\sqrt{2}})] \geq P_g, \tag{4.8}$$

$$\mu < x < \mu + \sigma\sqrt{2}. \tag{4.9}$$

In (4.2), for each substrate link uv, the embedded data rate is upper-bounded by a portion of the residual substrate capacity. The value $R \in \{R_h, R_m, R_l\}$ is the congestion ratio ((65)). Constraints (4.3) and (4.4) ensure that, given the source and destination substrate nodes, the

outgoing flow from the source node m and the total incoming flow in the destination node d is equal to the required data rate, with a negative sign for the latter one. For all intermediate nodes, (4.5) states that the sum of incoming flows has to equal the sum of outgoing flows (flow's conservation law). Equation (4.6) states the integrity constraint for the flow variable and constraint (4.7) guarantees the absence of loops. Unlike the traditional VNE approaches, we additionally introduce constraint (4.8) as a new measure to control the actual USP. In particular, (4.8) states that the probability of the actual traffic is lower than the assigned optimized data rate x (i.e., integration of the probability density function of the random variable X, over the interval [0, x]) should be greater or equal to the minimum guaranteed USP. Since the random variable x is within the erf function, the explicit formula of constraint (4.8) is obtained under the condition that the argument of the erf function is in the interval $[-1, 1]$. In addition, it should be convex, so the interval is further restricted to $[0, 1]$. This is why we introduce the additional constraint (4.9).

4.3 Statistics collection-aware VNE

Problem (4.1) is a mixed binary non-linear programming problem due to the binary VNR assignment variables and the non-convex objective function. In this section, we will first transform the original problem into a difference-of-convex (DC) form, which will be solved via an iterative algorithm. Subsequently, we relax the optimization problem presented in DC form using a proper penalty function that removes the binary constraint (4.6) and increases the scalability of the solution.

4.3.1 Difference of convex

The expression presented in (4.1) and in constraint (4.2) is not convex due to the multiplication of the flow variable times the assigned data rate variable. We will transform them into a preferable format which can be efficiently solved by DC. By using the equivalent representation of a product:

$$2xz_{uv} = (x + z_{uv})^2 - (x^2 + z_{uv}^2), \tag{4.10}$$

the objective function can be subdivided into two terms. The first one is already convex, while the second one is concave. To tackle this challenge, we will employ the first order of

approximation for the second term. In particular, for any feasible values $\bar{x}, \bar{z}_{uv}$, we can have following linear approximation:

$$(x^2 + z_{uv}^2) \simeq (2x\bar{x} - \bar{x}^2) + (2z\bar{z}_{uv} - \bar{z}^2). \tag{4.11}$$

By using (4.11) in the objective function, the original optimization problem (4.1) can be approximated as follows:

$$\min_{z_{uv},x} \frac{1}{|E_s|} \sum_{uv \in E_s} \frac{(x + z_{uv})^2 - 2x\bar{x} - 2z_{uv}\bar{z}_{uv} + \bar{x}^2 + \bar{z}_{uv}^2}{2(c(uv) + \epsilon)}. \tag{4.12}$$
$$\texttt{s.t.} \quad (4.2) - (4.9).$$

It is observed that constraint (4.8) can be equivalently converted into a linear constraint, since the function erfc() is monotonically decreasing. Therefore, one can readily solve the approximated problem (4.12) by standard Mixed-Integer tool since the objective function is convex and all constraints are linear. As a result, the solution of the original problem (4.1) can be obtained via a sequence of solving the approximated problem (4.12), in which the solutions of the i-th iteration $x_\star^{(i)}, \bar{z}_{\star,uv}^{(i)}$ are used as the feasible values at the $(i + 1$-th iteration, i.e., $\bar{x} \leftarrow x_\star^{(i)}$, $\bar{z}_{uv} \leftarrow \bar{z}_{\star,uv}^{(i)}$. Finally, the obtained solutions provide link mapping from source m to destination d as specified in the decision variable vector z.

4.3.2 SCA-VNE: Statistics Collection-Aware VNE algorithm

The pseudo-code of SCA-VNE is introduced in Algorithm 3. As mentioned earlier, the simulation is subdivided into time slots. At each time slot, a set of operations is performed, in the order as explained in Algorithm 3. Initially, in case of expired VNRs, the substrate resources will be released. For any mapped VNRs, the SDN controller checks and saves the current utilization. If the analyzed VNR's traffic is below θ and this has been registered for t_{exp} time slots, then the VNR is considered as expired. Otherwise, the counter for that VNR is increased by 1. In addition, if already t_{coll} samples have been obtained for the selected VNR, the VNR is included in the vector "to jointly optimize". Then, the new arrived VNR(s), if any, are initially mapped with DiVNE (57), restricted to the load balancing objective. We underline that we can not apply our model as initial mapping of any arrived VNR because the model needs to have a collected statistics. This is why we use a traditional VNE algorithm,

such as DiVNE, which is purely based on initial demanded resources. If the solution is found, the current substrate network resources are updated, and the traffic of the VNR starts to be transmitted. Otherwise, the VNR is inserted into the failed VNR(s) vector. This is done to initially assign a path, where the traffic can start to flow. The jointly optimization is performed later on, when t_{coll} samples of traffic have been collected, and, therefore, the actual value of traffic can be used to jointly optimize the assignment. This is what happens in the following for cycle, where, for all the VNRs which satisfy the condition of enough collected samples, the proposed joint optimization algorithm is proposed. All these described processes take a certain interval of time t_{comp}. For each time slot, given the fixed duration to t_{slot} seconds, in the interval $t_{slot} - t_{comp}$, the system will try to re-embed the failed VNRs, starting, once again, from the highest priority ones. This is important especially in congested scenarios, to try to minimize the delay experienced by the highest priority VNRs.

4.3.3 SCA-R: LP Relaxation of the SCA-VNE Formulation

In the previous subsection, we have presented the SCA-VNE formulation as a joint optimization and we have also showed that a non convex solution was transformed in a convex solution, using an iterative process. However, the exponential complexity of MBLP formulation limits the scalability of SCA-VNE. Figure 4.1 shows the exponential growth of the average computing time of SCA-VNE with the increasing number of the substrate nodes. This is due to the binary constraint (4.6) of the flow variable. For this reason, we propose a relaxed version of SCA-VNE, named SCA-R algorithm. SCA-R is obtained by the relaxation of the binary flow variable in constraint (4.6) to continuous values. To do that, we introduce the penalty function

$$P(\boldsymbol{z}_{uv}) = \sum\nolimits_{uv \in E_s} (z_{uv} - z_{uv}^2), \tag{4.13}$$

to penalize the values of the flow variable, far from the extremes 0 and 1, where $\boldsymbol{z}_{uv} \triangleq \{z_{uv}\}_{\forall uv}$. Since (4.13) is not convex, we introduce the first order approximation which results as

$$P(\boldsymbol{z}_{uv}; \bar{\boldsymbol{z}}_{uv}) = \sum\nolimits_{uv \in E_s} z_{uv} - 2 z_{uv} \bar{z}_{uv} + \bar{z}_{uv}^2, \tag{4.14}$$

where $\bar{z}_{uv}$ is any feasible solution. With the addition of $P(\boldsymbol{z}_{uv}; \bar{\boldsymbol{z}}_{uv})$ to the objective

Algorithm 3: SCA-VNE Algorithm

Initialization
Substrate network G_s, E_s, $c(uv)$ and $p(uv)$ $\forall uv \in E_s$;
VNR parameters (required physical resources, arrival distribution, end-to-end nodes);
for $t \in [t_0, t_{sim}]$ **do**
    save current status of substrate network;
    for *expired VNR(s)* **do**
        release assigned resources;
    end
    for any *mapped VNR(s)* **do**
        check data rate usage with SDN traffic statistics collection;
        save current utilization;
        if $x < \frac{\theta * \mu}{100}$ **and** $t_{count} = t_{exp}$ **then**
            add to *expired VNR(s)*;
        else
            increase t_{count} by 1;
        end
        if $t = t_a + t_{coll}$ **then**
            add in *to jointly optimize*;
        end
    end
    for any *arrived VNR(s)* **do**
        solve load-balancing (57) (starting from the highest priority VNR(s));
        if *successful* **then**
            update current network resources;
            send link mapping and demanded data rate to SDN controller;
        else
            add to *failed VNR(s)*
        end
    end
    for VNR(s) in *to jointly optimize* **do**
        solve (4.12) s.t. (4.2) - (4.9);
        if *successful* **then**
            update substrate residual capacities with optimized data rate x;
            send x to SDN controller;
        else
            add to *failed VNR(s)*
        end
    end
    for any *failed VNR(s)* **do**
        solve load-balancing as in (57) starting from the highest priority VNR(s);
        if *successful* **then**
            update current network resources;
            send mapping and assigned data rate to SDN controller;
        else
            try the subsequent VNR (if any)
        end
    end
end

function (4.12) of SCA-VNE, the final objective of SCA-R at the i-th iteration can be written as:

$$\begin{aligned} \min_{z_{uv},x} \quad & \frac{1}{|E_s|} \sum_{uv \in E_s} \frac{(x+z_{uv})^2 - 2x\bar{x} - 2z_{uv}\bar{z}_{uv} + \bar{x}^2 + \bar{z}_{uv}^2}{2(c(uv)+\epsilon)} \\ & + \gamma P(\boldsymbol{z}_{uv}; \bar{\boldsymbol{z}}_{uv}) \\ \texttt{s.t.} \quad & (4.2) - (4.5), (4.7) - (4.9). \end{aligned} \tag{4.15}$$

where $\gamma > 0$ as the penalty weight parameter. We observe that problem (4.15) is convex, hence can be efficiently solved by standard method. Compared with the SCA-VNE algorithm in the previous subsection, the proposed relaxed algorithm SCA-R does not have to deal with the binary constraint for the flow variable. We underline that, while the relaxed SCA-R has a relevant impact on the computing time, the difference in USP performance, as highlighted in Figure 4.2, is not dependent on the number of substrate nodes. Accordingly, the relaxed version SCA-R worsens the average USP of $\sim 1\%$ with respect to the integer version SCA-VNE.

4.3.4 Complexity Analysis

In this subsection, we analyze the worst-case scenario for the solution of the proposed algorithm SCA-R. For each iteration, SCA-R algorithm runs a convex optimization problem in (4.15), including $|E_s|+1$ scalar variables and $2|E_s|+|N_s|+2$ linear constraints. As a result, the per-iteration computation complexity required to solve (4.15) is (94) $\mathcal{O}(|E_s|+1)^3(2|E_s|+|N_s|+2))$. Given N_i as the number of iterations to reach a (locally) optimal solution, the global complexity to solve (4.15) is $\mathcal{O}(N_i|E_s|+1)^3(2|E_s|+|N_s|+2))$.

4.4 Simulation Results

In this section, we describe the simulation setup and we provide several performance evaluations to initially set problem parameters such as interval time slot duration and congestion ratio values for different priority VNR. Some trade-offs are discussed.

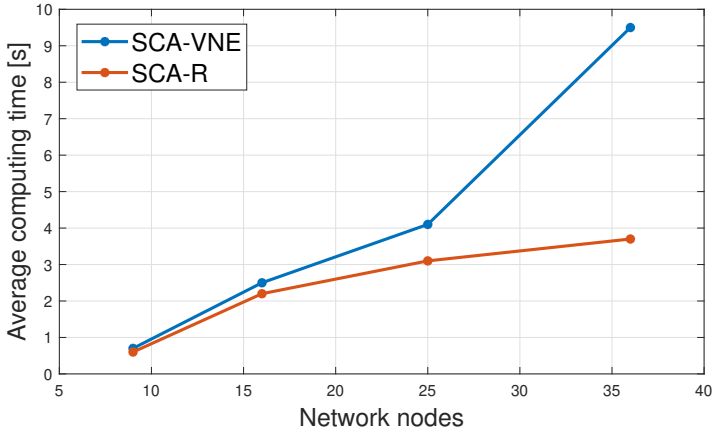


Figure 4.1: Comparison of average computing time vs network nodes between integer formulation (SCA-VNE) and relaxed one (SCA-R)

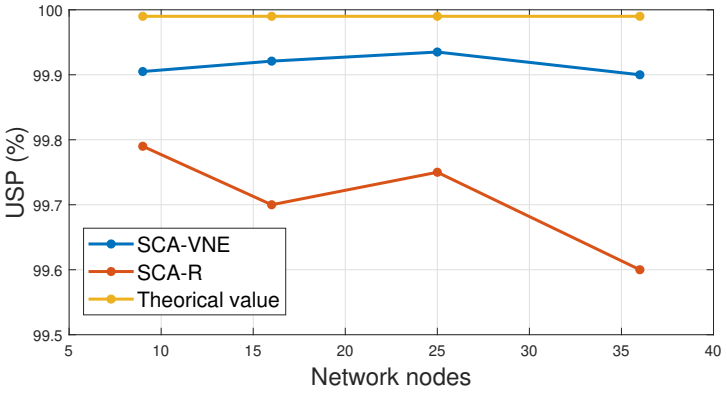


Figure 4.2: Comparison of average USP vs network nodes between integer formulation (SCA-VNE) and relaxed one (SCA-R)

4.4.1 Simulation setup

The substrate network is a grid of 16 nodes (SDN-enabled switches) and we simulate VNRs of different priorities. While considering a static network might seem a strong assumption, we clarify that, due to the complexity of the problem and the testbed, our intent in this work is to present the optimization and the logic behind the proposed mapping algorithm. Considering dynamic links would have further complicated the equations with the risk of confusing the reader, as this is a novel approach. However, we consider the interval time slot duration as low as 5-15 s. to reduce the impact of topology changes on the assumption of static network. As highlighted in Figure 4.3, among the services of beyond 5G and 6G networks are expected to be very heterogeneous. The same integrated non-terrestrial/terrestrial substrate network might accommodate emergency services (from IoT devices to Emergency vehicles), broadband and broadcasting accesses or very low priority and resource demanding services such as earth observations. This intends to motivate the reasons to consider several priority services with different QoS requirements. The VNRs are randomly generated between any pair of hosts with equal random probability (33.3 %) to be either high, medium or low priority requests. We initialize $\lambda = 1$ VNR/time slot. Note that we are generating high values of resource requirements and for a long period of time to congest the network after a few tens of time slot, considering that the congested scenarios are where the proposed algorithm shows the higher gain.

VNRs are differentiated on 3 different priority levels as follows:

- High priority e.g. the emergency connections;
- Medium priority e.g. the backhaul link for broadband connectivity;
- Low priority e.g. the earth observation missions.

For each level, we define $P_g = 99.99\%, 97\%$ and 90% for high, medium and low priority VNRs, respectively. We summarize the remaining parameters in Table 4.2.

4.4.2 Duration of time slot

In this subsection, we analyze the impact of the single time slot duration t_{slot}. Unlike other parameters such as congestion ratio or USP, t_{slot} affects not only the acceptance probability but, mainly, the average queuing delay. In fact, in each time slot, as described in Algorithm 3,

Table 4.2: Simulation setup parameters

Parameter	Value
Simulation time	1 hour
Initial substrate capacity c(uv)	10000 bps
Required datarate b	$\in$ [1000, 4000] bps
Lifetime L	$\in$ [100,150] time slots

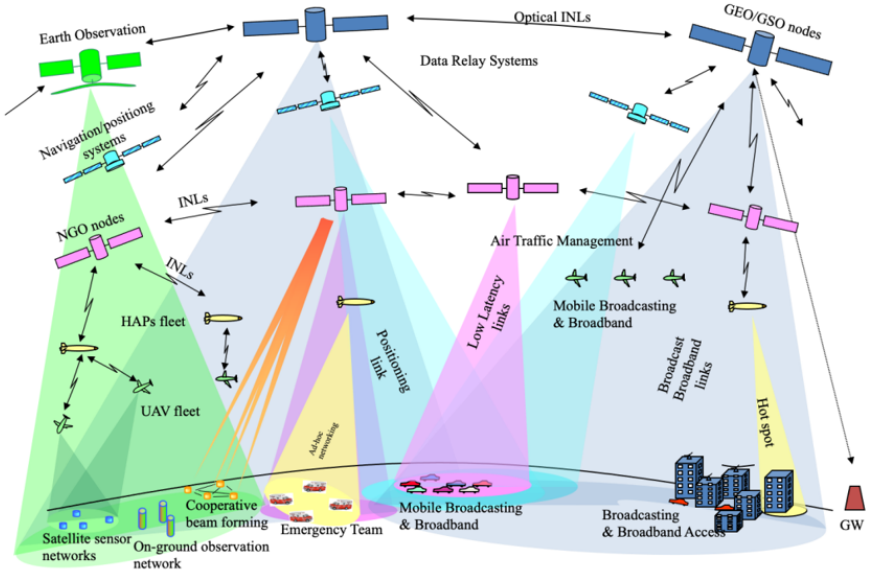


Figure 4.3: 6G Integrated Non-Terrestrial Networks (95)

not only the embedding for the new arrived VNRs is computed, but additional operations are performed, such as checking whether any VNR has expired, jointly optimizing any mapped VNR which traffic has been sufficiently collected for and trying to run again algorithm (57) for those VNRs which have failed the previous time slots. This latter operation is performed starting from the highest priority VNRs until the duration of the time slot is over.

Clearly, on one side, the longer the time slot, the more the probability to try to embed more previously failed VNRs (if any). However, on the other side, if one VNR fails to be mapped, it will have to wait the following time slot to try to be mapped again. This corresponds to a waiting time of at least t_{slot}. If the network load is low, this effect is not relevant because the system will usually manage to map all VNRs as soon as they arrive or few failed VNRs will be present each time slot. However, this might not be obvious when the load increases because the probability of having failed VNRs to be re-mapped increases as well.

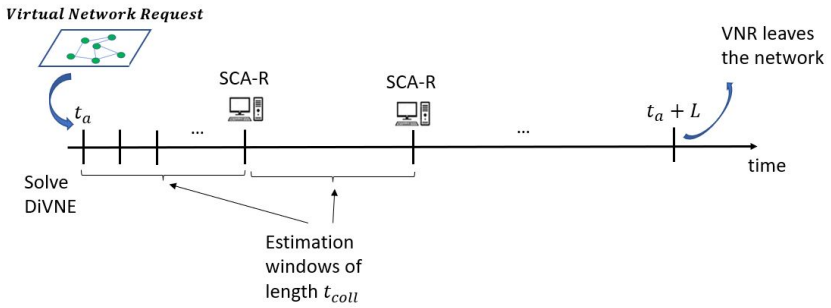


Figure 4.4: Lifetime of a VNR

To understand a reasonable value for t_{slot}, the average queuing delay should be considered. For the analysis we have taken three possible and meaningful interval values, 5, 10 and 15 seconds. In Table 4.3, we can notice that, as expected, independently on the priority, the higher the t_{slot}, the higher the average queuing delay. Note that the values are averaged over the accepted VNRs but, in general, the low priority VNRs are the most affected by a lower t_{slot} because they might suffer large delays.

After this analysis, $t_{slot} = 10s.$ is taken as the most reasonable choice for the following simulations.

Table 4.3: interval time slot duration impact on average queuing delay

t_{slot}	**High**	**Medium**	**Low**
5 s.	2.79 s.	4.51 s.	4.61s.
10 s.	2.9 s.	5.12 s.	4.99 s.
15 s.	3.549 s.	4.048 s.	5.4198 s.

4.4.3 Acceptance ratio vs time

The acceptance ratio evaluation is the performance used to analyze and obtain reasonable values of the parameters t_{coll} and t_{exp}. To better understand the value and the impact of these two parameters, we present in Figure 4.4 all steps each VNR goes through. Whenever a new VNR arrives, certain requirements are defined. In a real scenario, this can be seen as the resource requirements that a VNR has paid for. For the link mapping, the resource requirements are considered as the data rate and the USP. Initially, like the traditional approach, the VNR goes through an optimization algorithm where the path is optimized, given the required data rate as static. At this point, we solve the DiVNE (57).

If the mapping is successful, the SDN controller starts to collect data from the VNR. After

t_{coll} collected samples, the proposed joint optimized SCA-R, as mentioned in (4.15), is solved. The SCA-R is solved every collection window, i.e. every t_{coll} time slots. When the lifetime of the VNRs expires, the assigned substrate resources will be released. This is done because in the reality we know that the traffic is never static. In fact, after a set of collected samples, the SDN controller describes the traffic from each VNR, given the mean and the variance values. The number of the collected samples t_{coll} drives the first trade-off. The more the collected samples, the more precise the traffic estimation (higher USP) but the more the time where the VNR is assigned with a high value of data rate, which corresponds to less available resources for other VNRs. On the opposite, the sooner we get the traffic estimation, the sooner the joint optimization algorithm runs avoiding to waste substrate capacity, bringing to higher acceptance ratio.

Acceptance ratio, or acceptance probability, is a well-known performance analysis in VNE literature (54; 57; 64; 76; 77), defined as the ratio between the mapped VNRs and the arrived ones. It can be defined at each time, in case the simulation is subdivided into time slots and, by definition, is a number in the range $[0, 1]$. The second analysis, which drives the above-mentioned trade-off, is the USP. In fact, the lower t_{coll}, the worse the traffic estimation and the lower the USP. This effect is emphasized also by the fact that, during the interval of time where the traffic is being collected, the VNR is assigned with the maximum data rate, which means USP = 1 (from VNE perspective) in this scenario (with the assumption that there are no peaks of traffic higher than the required data rate). For this scenario, we have selected only one random type of VNR (medium) to avoid any worsening of performance due to mixing of priority mapping.

In Figure 4.5, the impact of t_{coll} on the trade-off between average acceptance probability is shown. In fact, on the x-axis, t_{coll} values are highlighted. On the left y-axis, we measure the average acceptance probability. The more the collected samples, the more the time spent with higher capacity consumption, so less available resources and therefore worse acceptance probability. On the other side, the more precise the traffic description, i.e. the system can find a more accurate optimized value, the higher the USP (right y-axis). This effect is emphasized also by the fact that for the interval of time where the traffic is being collected, the VNR is assigned with the maximum data rate, which means USP = 1 in this scenario (with the assumption that there are no peaks of traffic higher than the required data rate). This is the reason why the curve has a steep decrease.

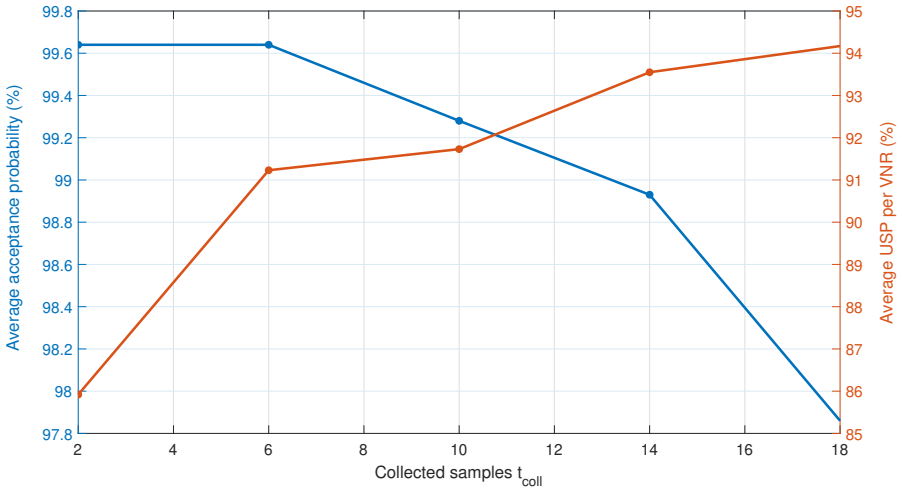


Figure 4.5: Average acceptance probability and Average USP vs collection window's length

After the first analysis, it looks like it is better to have $t_{coll} \geq 16$ time slots to avoid having low USP values.

It is worth underlining that the t_{coll} values that we have found is strictly dependent also on the type of simulated traffic. In this case, we are considering an easier scenario, where the VNR generates traffic described as a normal variable with average and variance equal to a percentage of the peak data rate (e.g. 90% of the peak data rate and 4%, respectively). We underline the fact that, depending on this choice, different results will be obtained. For example, if the traffic varies more, longer estimation times will be needed to collect reliable traffic description.

4.4.4 Congestion ratio impact and analysis

In this subsection, we analyze the impact of congestion ratio on the acceptance probability and we aim to set a reasonable value for congestion ratios for the three different priority values. As previously mentioned, congestion ratio is not a new concept. In fact, it has been introduced by authors in (65). If, on one side, congestion ratio is efficient to spread the traffic over the network and to leave available substrate capacity in case of need, on the other side, when the congestion ratio is too low, the acceptance probability is drastically reduced, hence, more free capacity is needed. Authors of CEVNE (65) have found out that a reasonable value for congestion ratio would be 0.945 in the presented scenario. This means that each substrate link can be filled up to 94.5 % of its capacity.

Table 4.4: Congestion ratio impact on acceptance probability

Congestion ratio scenarios	High	Medium	Low
$R_h = R_m = R_l = 0.945$	91.4 %	97.75 %	98.99 %
$R_h = 1$, $R_m = 0.98$, $R_l = 0.90$	95.65 %	94.19 %	92.23 %
$R_h = 1$, $R_m = 0.95$, $R_l = 0.90$	90.62 %	96.67%	83.87%
$R_h = 1$, $R_m = 0.95$, $R_l = 0.88$	96.88 %	100 %	93.55%
$R_h = 1$, $R_m = 0.92$, $R_l = 0.90$	96.88%	100 %	87.10%
$R_h = 1$, $R_m = 0.90$, $R_l = 0.85$	97.8 %	94.12 %	86.36 %

In this context, we apply the congestion ratio to the three types of VNRs. Initially, we use the same congestion ratio proposed in CEVNE, i.e. 0.945.

Results in Table 4.4 show that, considering the same congestion ratio for any type of VNRs is not efficient. In fact, the higher the priority of VNR (which corresponds to a higher USP), the lower the acceptance probability. Thus, in this scenario, the choice proposed for CEVNE is not efficient because, being the VNRs treated equally, high priority VNRs will be less likely to be accepted than the others, due to more need of capacity since the guaranteed USP is close to 100%. Clearly, the disadvantage introduced by the USP should be counteracted by an unbalanced congestion ratios, in favor of higher priority VNRs. It is worth underlining that these considerations hold in scenarios where the traffic demands have the same order of magnitude among the three types of requests (i.e. the traffic demand is randomly generated within the same range of values). In fact, in case of unbalanced situations, e.g. low priority VNRs with very low data rate, they would be even more likely to be mapped with respect to highly demanding requests.

We compare different combinations for R_h, R_m and R_l. The choice usually depends on the service level agreements (SLAs) and the load of traffic. Independently from the type of VNR selected, the first consideration is that the lower the congestion ratio, the lower the acceptance probability. In addition, using different values of congestion ratio depending on the priority, increases the probability of available capacity for higher priority VNRs. So the choice is whether to decide to penalize all VNRs at the same way, or to give importance for example to high and medium priority and accept a considerable lower values of acceptance probability for low priority VNRs. For example, if this is the case, considering $R_h = 1, R_m = 0.95, R_l = 0.88$ would make more sense. On the opposite, if we want to increase the chances for higher priority VNRs and decreasing the acceptance probability for low priority VNR is still within the SLAs, $R_h = 1, R_m = 0.90, R_l = 0.85$ would be a valid option, which offers a low acceptance

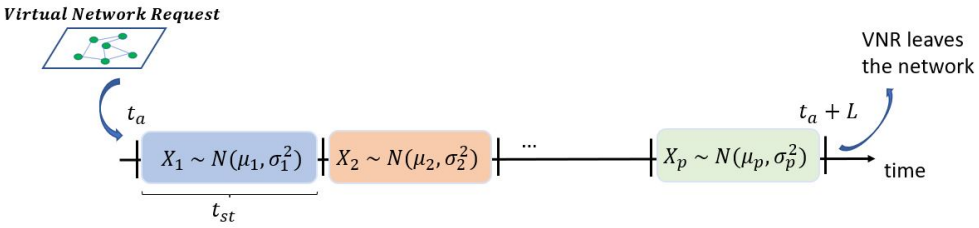


Figure 4.6: Non-stationary traffic

probability (86.36%) for low priority but the maximum for high priority (97.8%). For the following simulations, we take this latter option.

4.4.5 Average USP for non-stationary traffic

For the sake of simplicity, in the above initial analyses, we have considered a stationary traffic, i.e. static average and variance over time. However, given that the statistical distribution is by nature non-stationary, in this subsection we aim to test our algorithm in different scenarios of non-stationarity.

Let us assume that the VNR's traffic can be subdivided into epochs of length t_{st} where the statistic can be described with average μ and variance σ^2. As shown in Figure 4.6, let μ_i and σ_i^2 be the average and variance of the traffic during epoch i, respectively. We aim to test our algorithm with varying ratio $\frac{t_{coll}}{t_{st}}$, which considerably affects the final USP. We recall that t_{coll} is the number of collected samples to obtain the average and variance. Intuitively, the higher $\frac{t_{coll}}{t_{st}}$, the less prompt the reaction of the algorithm to quick changes of traffic statistics. On the opposite, the lower t_{coll} with respect to t_{st}, the more reactive the algorithm is. However, if t_{coll} is too small, the number of samples might not be enough to obtain a reliable statistical description. We run this experiment only for one type of VNR, high priority, because the same applies to the other types of VNRs. Figure 4.7 shows the impact of $\frac{t_{coll}}{t_{st}}$ on the USP. The theorical value represents the ideal case, i.e. when the estimation is always perfect and the traffic is stationary. When $\frac{t_{coll}}{t_{st}}$ is relatively low ($\frac{1}{3}$), the algorithm performs well because the statistics collected at the previous window, most likely apply to the current one, since for each t_{st}, three estimations are taken. The USP remains similar when $t_{coll} = t_{st}$ because the estimation is better due to more samples. However, a gain with respect to the previous case is not shown because the probability that the statistics between consecutive windows have considerably changed, i.e. previous estimations degrade the performance, compensates

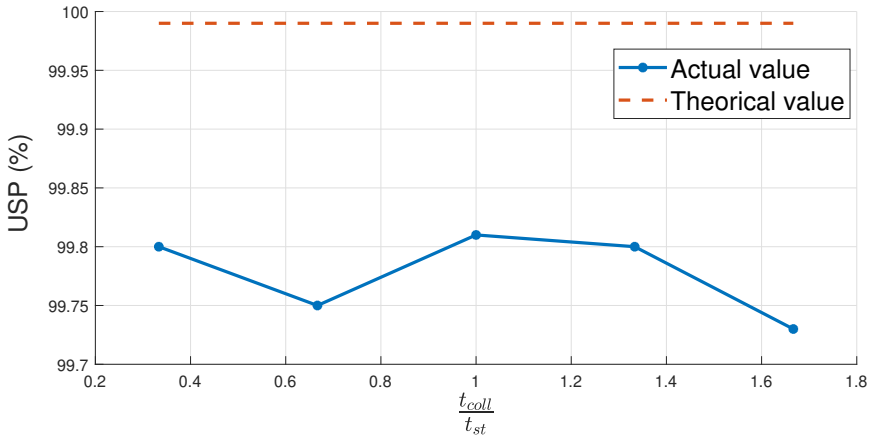
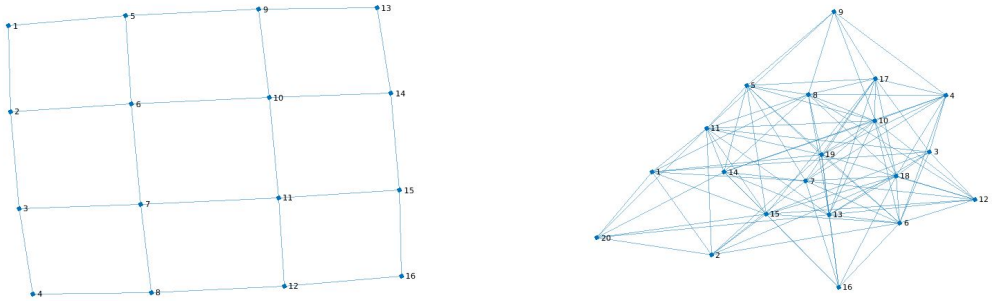


Figure 4.7: USP vs t_{coll}/t_{st}



(a) Scenario A (b) Scenario B

Figure 4.8: Comparison of network models

the final USP. Then, when $\frac{t_{coll}}{t_{st}} > 1$, the USP keeps decreasing because the algorithm is less reactive. This analysis highlights the efficiency of our algorithm when the collection window is comparable or lower to the interval of time where the statistics can be considered static.

4.4.6 Scalability of the substrate network

In this section, we test the algorithm with larger and more complex network. We take a substrate network composed of 20 nodes, with probability of being connected with each other equal to 0.5. We summarized in Figure 4.8 the network used for all the previous discussions, Scenario A, and the more complex network, Scenario B.

We aim to compare the three main KPIs, Table 4.5, to check the impact of the size of the substrate network. For this experiment, we considered only high priority VNRs in a low-load configuration mode. This is the reason why the acceptance ratio is in both cases

100%. This is done to avoid that the average delay is also impacted by the waiting time due to unavailability of the resources. It can be seen that the actual average outage in both cases is not as the expected one, i.e., 0.01%, due to error in the traffic description for limited samples. However, we should not expect that this is impacted on the network size. On the contrary, the average delay, which is the average computing time in this scenario, increases considerably with increasing network size.

Table 4.5: Comparison KPIs

Performance	**Scenario A**	**Scenario B**
Average outage	0.34 %	0.26 %
Average delay [s]	5.4	22.17
Acceptance ratio	100 %	100 %

As the results in Section 4.5 are obtained with the testbed, the configuration in Scenario B would generate so much complexity in the emulation side that the results are impacted from the hardware limitations of the tools, in particular network emulator and controller.

4.5 Testbed experiment results

After defining some relevant parameters, we compare the SCA-R to the following baseline approaches:

- CEVNE (65) restricted to link mapping (load-balancing approach);
- SP-1 (96);
- MIQCP (86), restricted to link mapping;
- SP-2, Shortest path with statistics collection.

We have chosen these 4 baselines for the following reasons. CEVNE and SP-1 are well-known VNE algorithms which belong to the categories of algorithms that do not collect any statistics and the embedding is executed purely on the required data rate for each VNR. MIQCP and SP-2, instead, represent the set of algorithms that consider also a traffic statistical description (considered to be known a priori) but without a joint optimization (probabilistic approach when optimizing the assigned data rate).

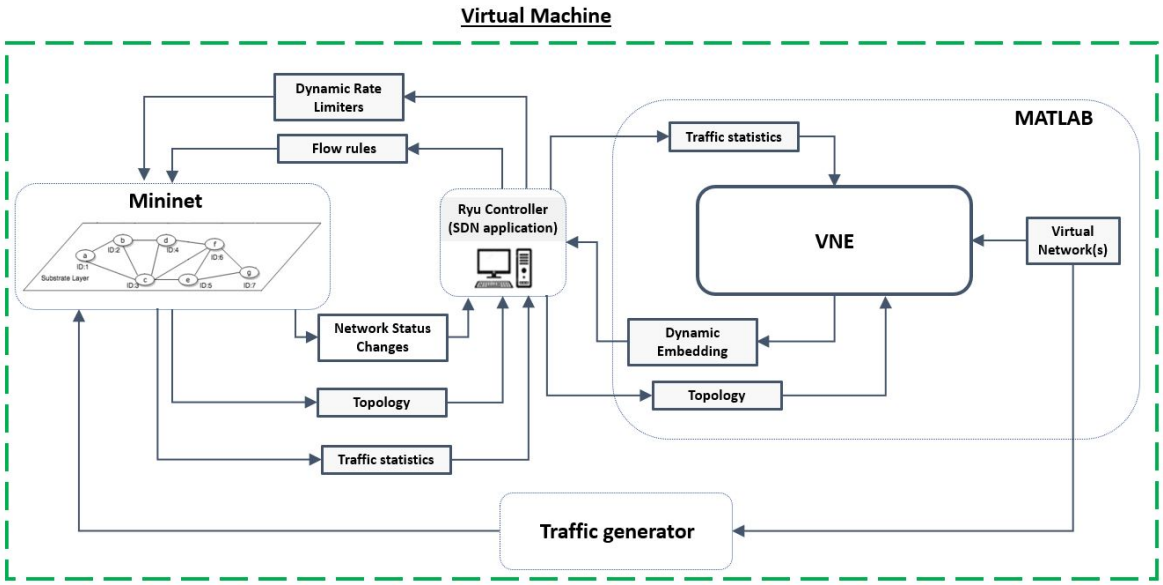


Figure 4.9: MIRSAT - Version 3

Table 4.6: Ostinato traffic description

L1	VLAN	L2	L3	L4	L5
MAC	Tagged	Ethernet II	IP	UDP	Empty Text

4.5.1 Testbed setup

For the previous results and discussion, we have used Matlab to run initial simulations to fix parameters such as collection time, congestion ratio and interval time slot duration. From now on, we validate SCA-R into a developed version of the testbed presented in (75). The schematic graph is shown in Figure 4.9. The testbed is composed by four main elements. The network emulator, Mininet, the SDN controller, RYU, the VNE algorithm running in Matlab and the traffic generator OSTINATO. We underline that the VNE might eventually run inside the SDN controller but, as the purpose of our testbed is to be tested with different VNE algorithms, we find it convenient to keep the two entities separated for a faster integration.

We use the traffic generator OSTINATO to simulate traffic demands, with a varying data rate over time depending on the traffic model we have selected. OSTINATO (36) creates strings of traffic with multiple choices for each ISO/OSI layer, starting from physical up to application layer. Table 4.6 summarizes the input settings, as defined in Ostinato. UDP traffic is used for each VNR, with vlan identifier to differentiate the VNRs. Then, the stream is initiated with the datarate and lifetime.

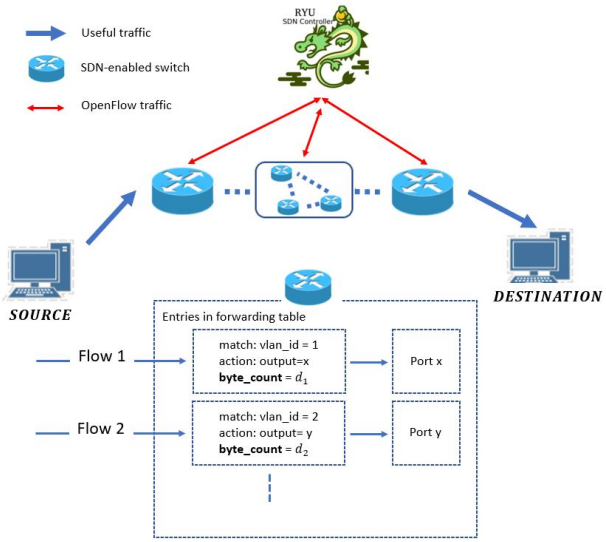


Figure 4.10: Traffic statistics collection

To collect statistics from every VNR, the process is the following. As shown in Figure 4.10, once the mapping is performed, the traffic flows from source to destination. The SDN controller, which is connected to all switches between source and destination, has access to the number of bytes transmitted for each installed flow of every switch ("*byte_count*"). So, given an interval of time Δt, the current data rate at time t in bps for the VNR j and for each link is passing through, is computed as:

$$\text{datarate(t)} = 8 \cdot \frac{d_j(t) - d_j(t - \Delta t)}{\Delta t}, \tag{4.16}$$

given $d_j(t)$ the transmitted bytes of VNR j until time t. It is worth underlining that, although the SDN controller computes the actual data rate for each link, we only select the data rate at the last switch, before the destination host, to have only one value per VNR.

After some offline analysis, $\Delta t = 5s.$ was taken as interval time for collecting the statistics. Lower Δt gave more unstable values which would have impacted the overall performance. On the other side, higher values would have slowed down the time slotting strategy of the VNE algorithm.

4.5.2 Acceptance ratio

In this subsection, the acceptance ratio comparison is shown. For all following experiments and results, we are using the parameters value obtained in the previous subsections, t_{coll},

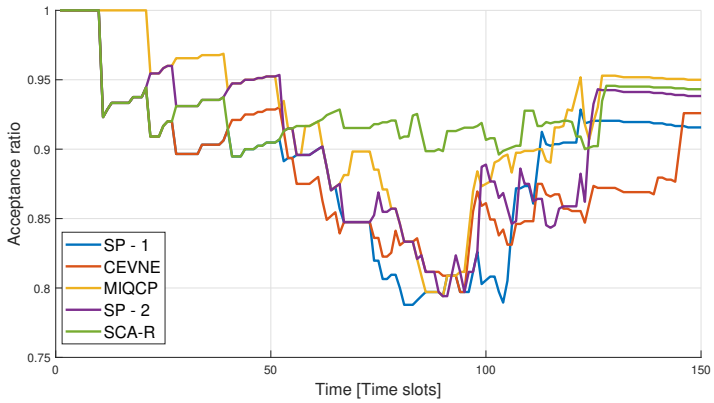


Figure 4.11: Acceptance ratio vs time

congestion ratios R_h, R_m and R_l and interval time slot duration t_{slot}. Two scenarios are considered. The first one (Figure 4.11) is longer and with higher available substrate capacity (20000 bps instead of 10000 bps) to decrease the load of the network. We plot the acceptance ratio over time for the four baseline algorithms and the proposed SCA-R. In this case, we that there is an initial part where the proposed algorithm performs even worse than most of the baselines. This is expected because SCA-R is using a congested ratio which reduces the acceptance ratio. However, as soon as the traffic is collected and the joint optimization is computed, more substrate capacity will be free, which brings to higher acceptance ratio (intermediate area). Afterwards, since we limit the number of generated VNRs to 100, the acceptance ratio will grow again since VNRs are also expiring, releasing capacity for the previously failed VNRs.

We also show that even better results are obtained with higher network load, as presented in Figure 4.12. Even in this case, at the beginning, the network is not congested and the efficiency of SCA-R cannot be appreciated because the probability that the VNR is not being accommodated is higher, since a lower portion of capacity can be used (congestion ratio). When the network is more congested, the efficiency of accumulating the statistics and jointly optimizing the embedding is visible and brings to higher acceptance ratio compared to baselines.

4.5.3 Average queuing delay

In this section we analyze the average queuing delay for each class of VNR. Firstly, we only consider SCA-R and we show that on a very low time scale, as presented in Figure 4.13,

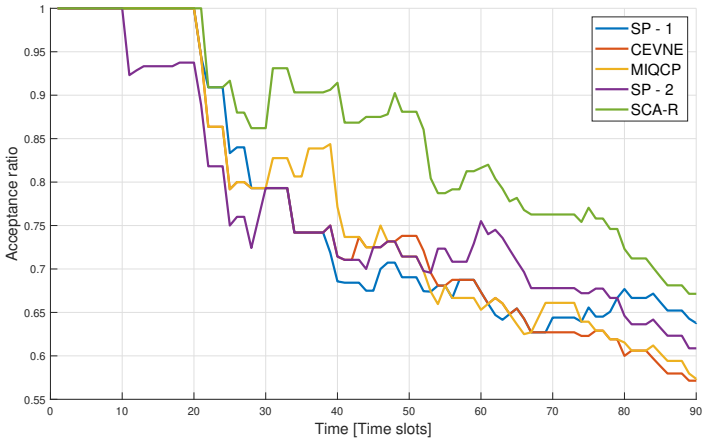


Figure 4.12: Acceptance ratio vs time in congested network

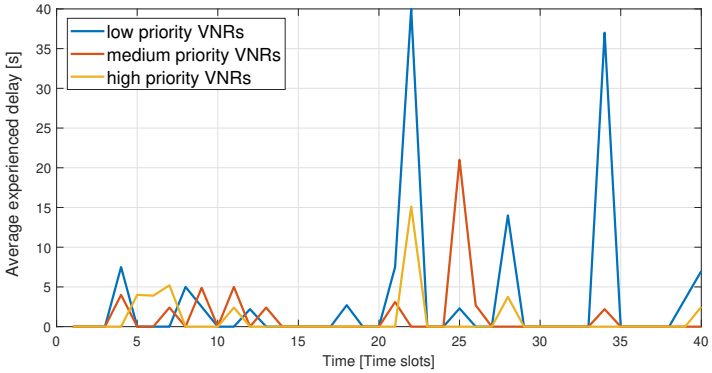


Figure 4.13: Average queuing delay

for each time slots, the lower the priority, the higher the average queuing delay. It is worth underlining that when in a time slot the delay of a high priority is higher than a lower one, this is because the highest priority VNR has failed to be mapped the previous time slots. Thus, its queuing time will be counting also the previous time slot duration. This is why we should look at the average queuing delay for each priority level. In addition, in the previous simulation, we showed that SCA-R improves, after a certain amount of time, the acceptance ratio compared to the baselines. As a consequence, on average VNR, independently of their priority, will have to wait more time before being embedded when baselines are considered. In fact, in the following table, we present the comparison of average queuing delay for the four baselines and for SCA-R, over a simulation time of 90 time slots, as presented in Figure 4.12. Due to decreasing acceptance ratio after 20 time slots, VNRs are more likely to wait. This is the reason why the value described below might seem high.

Table 4.7: Average queuing delay vs types of VNRs

Algorithm	High	Medium	Low
SP - 1	106.82 s.	276.14 s.	78.27 s.
CEVNE	198.74 s.	184.09 s.	112.06 s.
MIQCP	140.05 s.	152.64 s.	89.66 s.
SP - 2	154.90 s.	146.77 s.	125.27 s.
SCA-R	22.04 s.	115.48 s.	97.52 s.

Table 4.7 shows that when comparing the same priority level between SCA-R and baselines, the former always performs better. The fact that there are lower values for low priority VNRs with respect to medium priority is because the system is considering only the accepted VNRs, ignoring the ones which are not accepted. Thus, since the system starts to embed from the higher priority VNRs, few low priority VNRs are counted in this computation, which brings the average lower. This is also the reason why for low priority VNRs, it looks like SCA-R performs worse than some baselines. This is due to the fact that there are more accepted low priority VNRs than some baselines, and, therefore, the final average is counting more VNRs with respect to those baselines which have lower average queuing time.

4.5.4 Average USP

In this subsection, the average USP metric, computed as the percentage of time where the traffic is lower than the assigned data rate, is considered. It is worth noting that the first two baselines do not consider any probabilistic approach so, unless the traffic is higher than the maximum agreed level (which is outside the scope of this chapter), the USP is always equal to 1. For the baselines 3 and 4, there is a collection period, where the traffic is described as a random variable, which can bring to some incorrect values, but afterwards there is no joint optimization as SCA-R. Thus, if USP < 1, this is due to an error introduced by the number of analyzed samples. Even this matter is outside the scope of this chapter. For the above-mentioned reasons, it does not make sense to compare SCA-R to baselines, but we still provide a brief description of the results achieved. For each class of VNR, SCA-R optimizes the assigned data rate based on a defined minimum USP. Thus, in this section, we check the results of the obtained USP, which should be close to the expected values P_g.

It is worth adding that the congestion ratio does not influence the USP since, if the constraint on available capacity (lowered by congestion ratio) is not respected, the network

Table 4.8: Average USP

	High	Medium	Low
Expected	99.99%	97%	90%
Real	97.7%	95.6 %	89.4%

will not be mapped. Table 4.8 shows the results for the SCA-R, for the 3 different priority values. The results are not exactly equal to the expected values due to the error, in the traffic description, introduced by the limited number of samples.

4.6 Summary

In this chapter, we proposed a formulation for link mapping in VNE with joint optimization of load balancing and assigned data rate (SCA-VNE). Unlike the traditional approaches for VNE which assign to each VNR the resource requirements, either static or dynamic, in a deterministic way, SCA-VNE jointly optimizes the embedding for each VNR with a probabilistic approach. In fact, instead of assigning 100% of resource requirements for 100% of the time, SCA-VNE optimizes the assigned data rate with the guarantee that the USP is kept under control and lower-bounded by the minimum guaranteed QoS. Minimizing the assigned data rate to each VNR, increases the acceptance ratio due to more available substrate capacity. The effect of traffic's non-stationarity on SCA-VNE has also been discussed, showing a worsening of the performance with an increase of non-stationarity. SCA-VNE is supported by SDN, which describes the real traffic sent. Due to the exponential complexity of SCA-VNE, we proposed the relaxed version, named SCA-R. We compared SCA-R to four baseline approaches, and we proved that, the further the time goes, i.e. the more the load over the substrate network, the higher the acceptance ratio of SCA-R compared to baselines. In fact, initially SCA-R uses the traditional approach since there are no statistics collected on the traffic. However, when statistics are collected, SCA-R can run the joint optimization problem and the advantage starts to be noted. We differentiate VNRs based on a priority, which not only sorts them when being embedded, but also differentiates them when guaranteeing a defined USP. In fact, the higher the priority, the higher the USP. We also used known techniques such as congestion ratio, to increase the probability of accepting high priority VNRs. SCA-R has also shown a lower average queuing time for all priority levels compared to baselines. We have finally shown that the average USP values are close to the expected

values for the 3 different priority levels. We plan to extend this work to an integrated satellite-terrestrial network, simulated in STK, with dynamic links and real-time mapping decisions based, not only on the current traffic demand, but also on the available network connections.

Chapter 5

SAST-VNE: A Flexible Framework for Network Slicing in 6G Integrated Satellite-Terrestrial Networks

5.1 Introduction

Contrary to the previous two chapters, due to the relevance of network slicing in a very dynamic environment, namely multi-layer satellite-terrestrial 6G networks (97), this chapter presents the complete VNE problem, namely node and link mapping, for an integrated satellite-terrestrial network.

The high hetereogenity, introduced by the multi-layer structure of the satellite networks and the difference in terms of stability of the links and on-board resources, considerably increased the complexity of managing traffic demands for such dynamic network. In this context, technologies such as SDN and NFV are used in combination with a full-stack orchestrator to support NS operation (98). NS is based on the concept of developing and virtualizing a layered substrate network where each slice is seen as an isolated subset of node and link resources (e.g., processing, storage and communication).

Despite the well-known broad separation of use cases provided by the International Telecommunication Union (ITU) and the Fifth Generation Public Private Partnership (5G-

PPP), 6G networks are expected to define with greater granularity the differentiation of use cases, based on demands and application (99; 100). The more specific the differentiation of the use cases, the greater the need for the infrastructure to be flexible. In fact, network virtualization became relevant because it allows the service provider to exploit the dynamic nature of traffic, increase the network usage and allocate more traffic demands. The application of VNE has been investigated for more than a decade and with the advent of 6G networks, this complex optimization algorithm has become difficult to solve. As this work focuses on network slicing, throughout the contribution, we use the term "slice" to refer to the generic VNR.

This chapter investigates the application of VNE as the main NS enabler for an integrated satellite-terrestrial network managed by a SDN controller with joint load-balancing and minimization of the resources to be migrated during the satellite handover or network congestions. The chapter presents a flexible framework that adapts the time complexity and the optimality of the solution to be found, depending on the latency constraints.

5.2 Related Works

Authors in (101) studied a near-optimal formulation for VNE with a static infrastructure. Thanks to a procedure of building an augmented graph with candidate nodes, they manage to provide a coordinated node and link mapping. Authors in (65), propose a similar approach with the difference that they reduce the maximum utilization of each link to 94.5 %. Although these are near-optimal solutions, some critical points can be highlighted, which need to be modified when applied to future transport networks. For example, candidate nodes are discovered only on the basis of geographical distance. However, it is relevant to take into account scenario-dependent values of the link between the candidate and virtual node, such as the Signal-to-Noise Ratio (SNR). Furthermore, the authors consider resources and networks as fixed over a period of time and do not include migrations of the mappings. Providing a framework that constantly monitors, and eventually changes the current mapping of the slices, is relevant to be investigated while studying a VNE algorithm because the dynamic nature of future transport networks brings to unavoidable handovers and a minimization of reconfiguration costs is essential for infrastructure operators. Recently developed VNE solutions for dynamic contexts are proposed in (78; 102; 103). Authors in (78) investigate the

embedding of Virtual Network Function (VNF) for satellite networks. However, the mapping of nodes and links of VNFs is uncoordinated which typically reduces the quality of the embedding. In (102), authors propose SN-VNE for a satellite network with low complexity, which makes it favorable to compute during handovers in a reactive way, at the expense of the load-balancing performance. In (103), authors formulated the VNE for an integrated satellite-terrestrial network with the objective of minimizing the migrations during handovers. In this case, authors simplified the problem to a single End-to-End (E2E) VNRs, instead of considering an entire slice. The previous mentioned works deal with the optimization of the problem without associating real constraints or applications to the slices. Few works in literature consider 6G use cases which are expected to coexist over the same substrate network. Authors in (104) study the VNE applied to 6G use cases but they only focus on the link mapping, taking the node mapping as already pre-defined.

Furthermore, authors in (105) deal with the problem of E2E network slicing for NTN (Non-Terrestrial)-Terrestrial networks. However, there is no study of the response to real-time situations, such as congestion, with minimization of reconfigurations during handovers. As already proposed in works such as (104; 106; 107), the application of VNE for a satellite-terrestrial integrated network is feasible thanks to novel technologies, such as SDN, which simplify the networking and make it flexible and reactive to changes. In (106), authors exploit the idea of creating a framework for network slicing in integrated satellite-terrestrial networks, with the focus on respecting the QoS requirements, such as bandwidth and E2E delay, of different slices. Our solution intends to improve the framework proposed in (106), with the generalization of the slices to multiple links, instead of a single E2E connection, and minimization of the re-configuration cost while handovers are required. The embedding of slices is often accompanied by a SDN implementation, such as in (107), where the authors implement a SDN application that manages the embedding of slices in the satellite-5G network. However, there is no optimization involved. Our aim is to present an SDN-oriented framework that can be easily interfaced with an SDN-based platform, such as the one proposed in (104).

To the best of our knowledge, no VNE framework has been presented before to support NS for 6G use cases in an integrated satellite-terrestrial network, with a joint optimization function of load-balacing and migration cost minimization focused on real-time network events in either a reactive or proactive manner.

5.3 Motivations

In the previous section, we described the current literature review of VNE and discussed its application for NS in satellite-terrestrial networks. The NP-hardness of VNE has brought trade-offs between complexity and performance. The main motivations of our work are summarized as follows:

- The use of a single VNE solution cannot efficiently handle future dynamic transport networks. In fact, there is a need for a flexible framework that gives higher priority to performance, i.e., higher computing time, when there is no strict latency constraint. On the contrary, a simpler VNE solution to provide a less optimal embedding when the application is critical.
- Some proposed works in the literature showed how to coordinate the node and link mapping by building an augmented substrate graph. However, candidate nodes are typically chosen only based on a pure 2D geographical distance, which does not reflect the nature of some communications.
- Instead of randomly generating VNRs in terms of resources' demand, it is relevant to test VNE algorithms with real-traffic requirements, where each slice can reflect a proposed use-case for the next generation of transport networks.
- The majority of the near-optimal solutions, due to their complexity, do not provide any minimization of migration's cost when a slice need to be moved because of lack of available substrate resources or due to handovers. This is relevant in future networks where their dynamic exploitation can require frequent migrations.

As we listed the main motivations of our work, in the following we highlight the respective contributions as follows:

- We provide a flexible framework to support the VNE application for integrated satellite-terrestrial networks. We propose SAST-VNE, a new formulation for VNE that provides a joint objective function with load-balancing and minimization of migration cost. SAST-VNE provides a coordinated node and link mapping with a common approach of building an augmented substrate graph;

- We propose a novel way of looking for candidate nodes in the node mapping strategy. While generally this is done via geographical location, we include additional features to improve the quality of the embedding such as the SNR. Furthermore, we run the candidate search on a 3D space instead of 2D;
- We apply SAST-VNE to the NS concept, where each VNR is seen as a real network slice with requirements coming from 6G use cases. A priority is assigned to each class of slice based on their time constraints. SAST-VNE manages the satellite handover for all classes in a proactive manner, minimizing the migrated nodes and links;
- We simulate a constant monitoring of the network with real-time analysis on whether a slice, or even only a smaller portion of it, is problematic, i.e., does not respect the QoS. Our framework minimizes the migration cost, i.e., number of migrated nodes and links, when the latency constraint of the slice allows it. For critical slices, a heuristic solution is found to minimize outage during migration.

This chapter is organized as follows. Section 5.4 formulates the problem. Section 5.5 presents the algorithm SAST-VNE and its relaxed version. Section 5.6 describes some initial simulations and Section 5.7 presents the performance evaluation. Finally, Section 5.8 concludes the chapter.

5.4 Network Model and Problem description

5.4.1 Substrate Network

We model the substrate network as a directed weighted graph $G_s = (N_S, E_s^t)$, where N_S is the set of substrate nodes and E_s^t the set of substrate edges at time t. Each substrate node $n_s \in N_S$ is associated with the available node-related resources (CPU, storage and memory) $R_N(n_s)$, the location $loc(n_s)$ on a three-dimensional space. Given u and v a pair of substrate nodes in N_S, the link e_{uv} is associated with the residual capacity $R_E(e_{uv})$.

5.4.2 Slice Request

As described for the substrate graph, we model each slice request as an oriented subgraph $G^v = (N^v, E^v)$. We assign the term "virtual" to the nodes and links of each slice G^v. Each virtual node n^v has a location $loc(n^v)$ and a demanded node-related resources $c_v(n^v)$. Each

virtual link i is assigned with the demanded datarate b and a priority p is assigned to each slice.

5.4.3 Problem Formulation

In this section, we include the objective function and the constraints of our problem. The objective function (5.1) jointly optimizes the load-balancing (capacity for links and available resources for nodes) and the differences between consecutive mappings, i.e., it reduces the cost of slice migrations. It is composed of four main terms. The first one computes the sum of the percentage of used capacity for each substrate link e_{uv}. The second one, similarly, is the sum of the consumption (in percentage) of the resources for each substrate node. The last two terms compute the differences in the node and link mapping between the current solution and the newly computed one.

We formulate the objective function as:

$$\begin{aligned}\min_{f^i_{e_{uv}}, x_{mw}} \quad & \frac{1}{|E^t_s|} \sum_{e_{uv} \in E^t_s} \frac{\alpha}{R_E(e_{uv}) + \epsilon} \sum_i f^i_{e_{uv}} \\ & + \frac{1}{|N_S|} \sum_{w \in N_S} \frac{\beta}{R_N(w) + \epsilon} \sum_{w \in N_{S'} \setminus N_S} x_{mw} c_v(m) \\ & + \frac{\gamma}{|E^t_s|} \sum_{e_{uv} \in E^t_s} |\mathbf{F^{t-1}} - \mathbf{F^t}| + \frac{\delta}{|N_S|} \sum_{e_{uv} \in E^t_s} |\mathbf{X^{t-1}} - \mathbf{X^t}|\end{aligned} \tag{5.1}$$

where $(\alpha, \beta, \gamma, \delta) \in \mathbb{R}^4_+$ are the weights for the link mapping, node mapping, difference between consecutive link and node mappings, respectively. In addition, we define $\mathbf{F^{t-1}}$, $\mathbf{F^t}$ as the link mapping matrices at time slot $t-1$ and t, respectively. Similarly, we define the node mapping matrices $\mathbf{X^{t-1}}$ and $\mathbf{X^t}$. It is important to underline that only $\mathbf{F^t}$ and $\mathbf{X^t}$ contain the variables f and x because the mapping in the previous time slot $t-1$ has already been defined. We summarize the optimization variables in Table 5.1. In the following, we present the constraints of SAST-VNE.

To facilitate reading, we highlighted the variables in bold.

$$R_N(w) \geq \mathbf{x_{mw}} c_v(m), \forall m \in N_{S'} \setminus N_S, \forall w \in N_S, \tag{5.2}$$

Table 5.1: Variables in eq. (5.1) - (5.12)

Variable	Explanation
$f^i_{e_{uv}} \in \mathbb{R}_+$	flow variable to indicate if the virtual link i is embedded in $e_{uv} \in E^t_s$
$x_{e_{uv}} \in \{0,1\}$	node mapping variable

$$\sum_i (\mathbf{f^i_{e_{uv}}} + \mathbf{f^i_{e_{vu}}}) \leq R_E(e_{uv})\mathbf{x_{e_{uv}}}, \quad \forall i, \forall (e_{uv}) \in E^t_s, \tag{5.3}$$

$$\sum_{w \in N_{S'}} \mathbf{f^i_{uw}} - \sum_{w \in N_{S'}} \mathbf{f^i_{wu}} = 0, \quad \forall i, \forall u \in N_{S'} \setminus \{s_i, d_i\}, \tag{5.4}$$

$$\sum_{w \in N_{S'}} \mathbf{f^i_{s_i w}} - \sum_{w \in N_{S'}} \mathbf{f^i_{w s_i}} = b(e_i), \quad \forall i, \tag{5.5}$$

$$\sum_{w \in N_{S'}} \mathbf{f^i_{d_i w}} - \sum_{w \in N_{S'}} \mathbf{f^i_{w d_i}} = -b(e_i), \quad \forall i, \tag{5.6}$$

$$\sum_{w \in \Omega(m)} \mathbf{x_{mw}} = 1, \quad \forall m \in N_{S'} \setminus N_S, \tag{5.7}$$

$$\sum_{m \in N_{S'} \setminus N_S} \mathbf{x_{mw}} \leq 1, \quad \forall w \in N_S, \tag{5.8}$$

$$\mathbf{x_{e_{uv}}} = \mathbf{x_{e_{vu}}}, \quad \forall (e_{uv}) \in E^t_s, \tag{5.9}$$

and then the variable constraints:

$$\mathbf{f^i_{e_{uv}}} \geq 0, \quad \forall i, \ \forall u, v \in N_{S'}, \tag{5.10}$$

$$\mathbf{x_{e_{uv}}} \in \{0,1\}, \quad \forall i, \ \forall u, v \in N_{S'}, \tag{5.11}$$

Constraint (5.2) makes sure that among the candidate substrate nodes, only the ones with enough available resources are selected. Equation (5.3), for every virtual link, selects the substrate link with enough capacity. Then, for each virtual link i, the flow's conservation law is enforced by constraints (5.4)-(5.6). Constraint (5.7) sets the maximum number of

selected substrate nodes, among the candidates, equal to 1 while constraint (5.8) ensures that each substrate node is not selected by more than one virtual node. In constraint (5.9), for each substrate link, the x variable is set to 1 as long as there is a flow in any direction on that link. Finally, constraint (5.10) sets the positivity of the flow variable and constraint (5.11) forces the integrity of the node variable x.

5.4.4 Linear Programming relaxation and rounding technique

As the problem formulated in the previous section is a Mixed Integer Programming (MIP) which is known to be computationally complex, we provide a relaxed version of the problem. Accordingly, we relax the constraint (5.11) so that it becomes as follows:

$$0 \leq \mathbf{x_{e_{uv}}} \leq 1, \quad \forall i, \ \forall u, v \in N_{S'}. \tag{5.12}$$

5.5 SAST-VNE Framework

The results of the optimization problem (5.1) in Section 5.4 provide the optimal coordinated node and link mappings to meet the demands. The logic behind the coordinated node and link mapping is as follows. For each virtual node, the cluster is defined with the candidate substrate nodes to host it. For each candidate node, a fractional value of $x_{e_{uv}}$ is obtained. For each metalink (from candidate to virtual node), the flow variable is computed, normalized to be in $[0, 1]$ and multiplied by $x_{e_{uv}}$. Finally, we select the substrate node based on the highest value of the obtained products, while the other candidates are discarded. This process is repeated for each virtual node. Once the node mapping is defined, the link mapping is computed sequentially by solving the Multi-Commodity Flow (MCF) algorithm for each link. In this section, we present the structure of our proposed SAST-VNE framework, see Algorithm 4, and we analyze the complexity of (5.1).

5.5.1 Description

The pseudo-code of algorithm 4 is described in the following. There is an initial step where the algorithm requires to set parameters, such as the substrate network features (capacity for the links, time description of network behavior, node-related resources). Then the slice generation mechanism is initiated. Once the substrate network and slices are randomly generated,

the simulation with a time-slotted strategy runs. For each time slot, some operations are performed. Initially, the KPIs of any mapped slice(s) are checked. If one or more KPIs are not respected or the slice will undergo satellite handover in the following time slot, the slice is considered *problematic* and will be inserted into a queue to be analyzed. In case of any new arrived slice(s), the embedding according to D-ViNE is computed. Lastly, for any problematic slice, our VNE algorithm runs. In case it succeeds, part or the full slice will be moved toward the new embedding. For this last cycle, to be more precise, depending on the priority of the slice and on its situation (handover or KPI failure), different versions of the VNE are run. In fact, if a high priority slice is impacted by congestion, each affected link is re-mapped with shortest-path algorithm, given the source and destination of the affected link(s), to speed up the embedding process and provide connectivity in low time. On the contrary, lower priority slices are mapped using our proposed algorithm, which minimizes the migration cost.

5.5.2 Complexity Analysis

To check the complexity of SAST-VNE, it is enough to analyze the solution of the complex algorithms, i.e., equation (5.1), VINEYard and shortest-path. As the equation 5.1 is a more complex version of VINEYard and the shortest-path, due to the introduction of two main elements in the objective function, it is the element that dominates the time complexity. The time complexity of (5.1) is the sum of the complexity of the first two main elements, which can be described as $O((|E^{S'}|(1+|E_V|))^{3.5})$, plus the complexity of the third element which is similar to the objective proposed in (103), restricted to the first 2 time slots, i.e., $O(g^{2\cdot(h-1)})$, with g the maximum degree of the substrate graph and h the path length, plus the complexity of the last element which is comparable to the number of permutations of $|N_V|$ elements among the set of $|N_S|$ elements, i.e., $O(\frac{|N_S|}{|N_S|-|N_V|})$.

5.6 System Model

In this section, we initially present the simulation setup and, then, we provide several performance evaluations to initially set some problem parameters such as the criteria to define and compute the augmented substrate graph (i.e., the expanded substrate graph with the candidate nodes) and the weights involved in the objective function. Therefore, some trade-offs

Algorithm 4: SAST-VNE

Initialization
Substrate network G_s, E_s^t, $c(e_{uv})$ and $d(e_{uv})$ $\forall e_{uv} \in E_s^t$;
Slices' generation process: arrival, graph G_V and node/link requirements);
for $t \in [t_0, t_{sim}]$ **do**
    save current status of substrate network;
    for any *mapped slice(s)* **do**
        for any virtual link **do**
            check KPIs;
        end
        save current utilization;
        if KPI(s) not respected **then**
            add to problematic slice
        end
        if handover in the next slot **then**
            add to problematic slice
        end
    end
    for any arrived slice(s) **do**
        sort based on priority;
        solve VINEYard;
        if *successful* **then**
            update current network resources;
        else
            add to *failed slice(s)*
        end
    end
    for any failed/problematic slice(s) **do**
        sort based on priority;
        if *high priority* **then**
            solve shortest-path for affected links;
        else
            solve (5.1) s.t. (5.2)-(5.10), (5.12);
        end
        if *successful* **then**
            update current network resources;
        end
    end
end

are discussed.

5.6.1 Simulation setup

The substrate network is simulated via STK (73) and is an integrated satellite-terrestrial network, as in (104), composed of 32 LEO satellites from Iridium constellation and one GEO satellite (SES-6) (see Figure 5.1). We assign a location to each substrate node according to the geographical position on a 3D space of dimensions 300x100x300. For the sake of simplicity, we consider the LEO satellites, when in Line-of-Sight (LoS), in a fixed location inside the 3D space. Specifically, we locate the LEO satellite on the plane xyz with z = 100, while we locate the GEO satellite at z = 300. One could argue that these values do not reflect the reality as the LEO satellites orbit at around 800 km while the GEO at 36000 km. However, as in our work we build the augmented graph based on the geographical distance, see equation (5.13), which is controlled in value by the parameter S, with the smallest difference in altitudes between terrestrial, LEO and GEO nodes, we avoid to consider very low values of S. The terrestrial components are the GEO gateways (69), LEO gateways and users and GEO users, which are clearly located at z = 0. As this work is intended to be an extended version of (104), we consider the same type of traffic differentiation as described in Table 5.2. However, unlike the previous work, we consider an entire slice, i.e., a set of interconnected nodes and links, instead of a single E2E connection. Each substrate node is initiated with available resources $\in [25, 45]$ while each link with available capacity = 400 for terrestrial links, 1000 for LEO links and 10000 for GEO. We intentionally assign less capacity to the terrestrial links to make sure the system chooses the satellite links with more likelihood so that our algorithm can be tested in a more dynamic environment.

Table 5.2: Slices description

Slice priority	**Application**
High - 1	emergency services, collision avoidance scenarios
Medium - 2	infotainment, IFC, earth observation
Low - 3	autonomous boat driving, device manufacturer updates

Each slice is randomly generated as a graph N^V, with the number of nodes as a random number in the interval [2, 5]. Then, we set the probability of every pair of nodes to be connected, equal to 0.5. If the randomization process produces a graph with nodes that are not connected, the slice is discarded as we intentionally aim at connected graphs. Each node

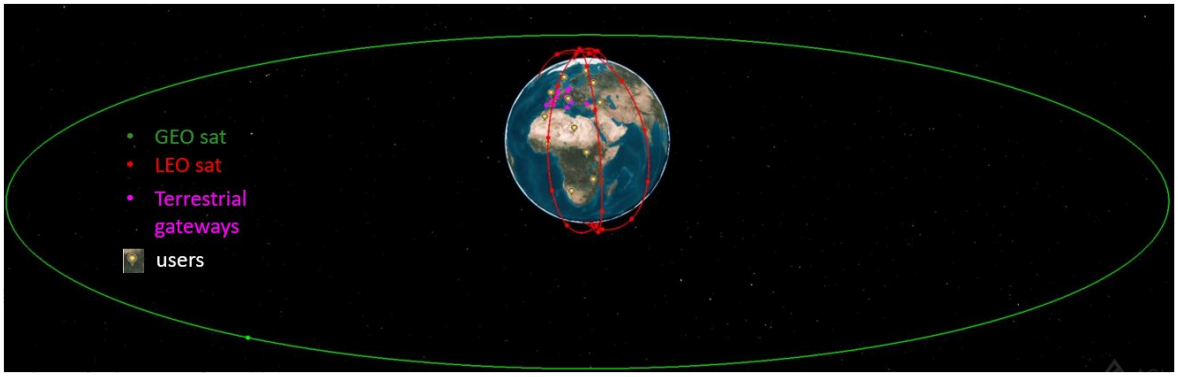


Figure 5.1: Combined GEO-LEO network

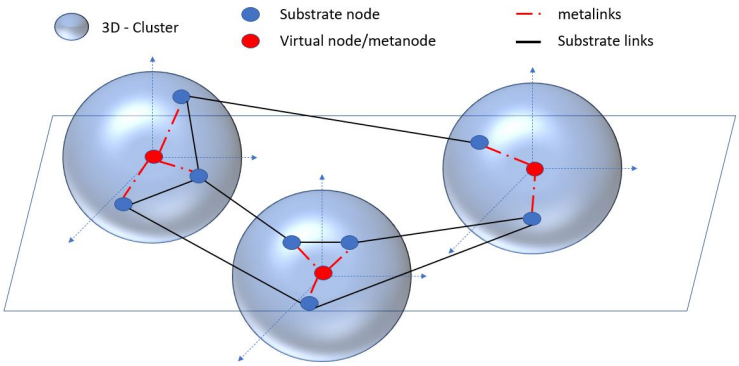


Figure 5.2: Definition of the augmented substrate graph

in E^V is located in a random position in the xyz plane (with z randomly $\in [0, 20]$ to simulate traffic demands from terrestrial users or plane), and it demands for random resources $\in [2, 6]$. Each link in E^V demands a datarate $\in [50, 150]$.

5.6.2 Creation of augmented graph

As already proposed in (101), when embedding a slice, an augmented graph (Figure 5.2) is suggested. It is built starting from the initial substrate graph which is expanded with *metanodes*, i.e., the nodes to be embedded and *metalinks* as the link between *metanodes* and the candidate subtrate nodes to host the *metanodes*.

The selection of the candidate nodes plays a key role in the complexity of the problem because the larger the selection of the candidates, the higher the number of links and, consecutively, the higher the complexity. As shown in (101), and reported here in Section 5.5.2, the complexity of the relaxed version is linearly dependent on the number of links in the

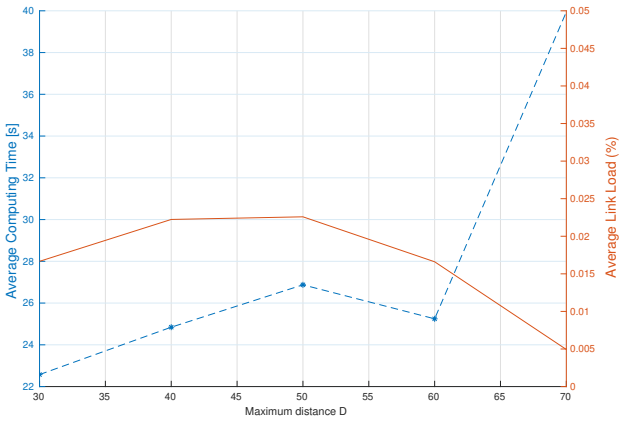


Figure 5.3: Time complexity and Load-Balancing vs minimum distance

augmented substrate graph. Authors in (101) select the candidate nodes based on the geographical location on a 2D space. As anticipated in section 5.3, our work brings several novelties in the definition of the candidate nodes. Firstly, we build a 3D structure, as the nature of integrated terrestrial-satellite networks foresees. Furthermore, we assign to each link between a generic metanode A and a substrate node B, a signal-to-noise ratio coefficient $S \in [0, 1]$ that multiplies the euclidean distance $d(A, B)$ as explained in (5.13).

$$d(A, B) = S \cdot \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2}. \tag{5.13}$$

The parameter S is used to modify the distance between the candidate node and the virtual one based on the current SNR. This is used to make the scenario more realistic. For instance, a terrestrial node which is physically closer to the virtual node on ground, eventually can have a worse link quality, e.g., due to terrestrial obstacles, compared to the satellite link. We randomized the value of $S \in [1, 4]$ for the terrestrial nodes, while we select a more constant interval for the LEO links at $S \in [0.5, 1]$. We fix $S = 0.2$ for the GEO link as we assume it to be stable. With these premises, this subsection aims to fix a priori a reasonable value of the maximum distance D between the metanode and the substrate nodes at which the substrate node can be considered a candidate one. It is worth noting that this selection is driven by the trade-off between the time complexity and performance, as shown in Figure 5.3, and it is strictly linked to the chosen substrate topology.

In fact, as shown in Figure 5.3, the higher the maximum distance D, the higher the average time complexity. In this case, when D $>$ 60, the complexity grows exponentially. On

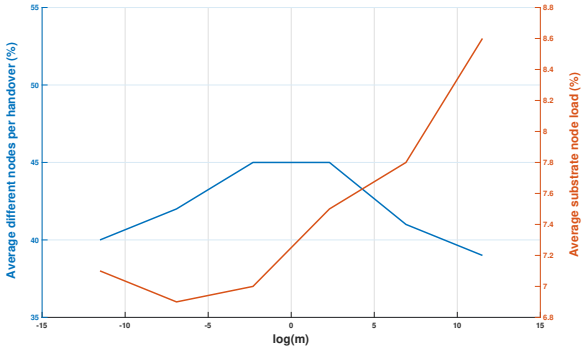


Figure 5.4: Average node migration and substrate node load vs cost factor

the other hand, the higher D, the more the candidate nodes, and thus, the easier the load-balancing. In fact, the average link load decreases as D increases. This preliminary analysis sets the maximum distance to be considered for candidate substrate nodes to 50. We also note that, when the substrate network is dense, the reduction of D does not automatically correspond to a reasonable decrease of complexity. As explained in Section 5.5.2, because the complexity is exponentially linked to the number of links of the augmented substrate graph, in a dense network, even with small D, the number of connected nodes can be very high. For this reason, we introduce a feature in SAST-VNE that limits the maximum number of candidate nodes when building the augmented substrate graph.

5.6.3 Weight parameters definition in objective function

In this subsection, we study the impact of the weight parameters introduced in (5.1) to properly set their initial values. Even in this case, a trade-off is highlighted. The weights α and β are assigned to the link and the node load balance, respectively. While γ and δ are for the link and node mapping differences between previous and new mappings, respectively. To properly select the values for the four weights, we initially check the order of magnitude of each of the four terms of (5.1). Intuitevely, all the terms are percentages as they are divided by the number of nodes or links. In this simulation, we fix the parameters as follows:

$$\alpha = \beta = 1 \tag{5.14}$$

$$\gamma = \delta = m. \tag{5.15}$$

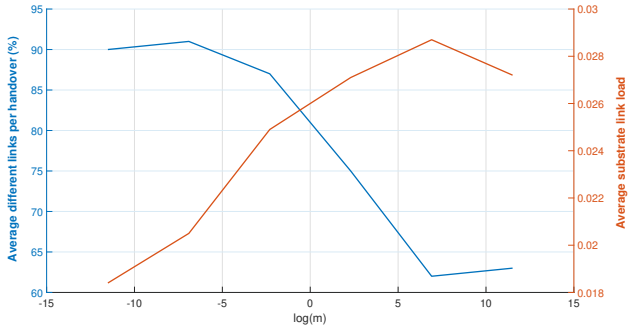


Figure 5.5: Average link migration and average substrate link vs cost factor

We test the SAST-VNE with different orders of magnitude for m. The aim is to find the suitable value to have a fair trade-off between load-balancing and low migration cost. We consider separately the cost of node migration and link migration. The average node migration is computed as the number of nodes that are migrated everytime there is a change in a slice. Then it is averaged over all slices. Similarly, the average link migration is computed. In Figure 5.4 we compare the average node migrations and the average network load vs. the cost factor m. Some comments can be highlighted. First, as expected, the overall trend of the performance measuring differences after a congestion, on average, decreases when the cost factor is increased, i.e., the weight in the objective function (5.1). The opposite happens for the average load of the substrate nodes, computed as the average among the nodes of their CPU capacity utilization during the simulation. It can also be noticed that the average substrate node load remains more or less constant. This is probably due to the fact that the congestion is created only in the links, and therefore, the chance that the nodes remain unchanged but different links are chosen is higher.

Similar considerations can be highlighted when comparing the average migration and load of the link, as shown in Figure 5.5. In this case, the pattern of both curves is closer to the expected one (cross shape) because we simulate the congestion over the links. Precisely, we randomly select 30 % of the links where the current mapped slices are accommodated, and set their capacity to 85 % of usage to almost always force the mapping over different substrate links.

To avoid having a too unbalanced solution, from these simulations, it looks clear that values of the cost factor close to unity can be taken. Clearly, a further trade-off may also be analyzed in case more priority should be given to node migrations rather than link migrations.

In other words, this would mean considering $\gamma \neq \delta$. As this would increase the computational cost, but, nevertheless, without bringing any relevant additional information, we can skip it in this work.

5.7 Simulation results

In the previous section we introduced the simulation setup and described some preliminary evaluations to start our SAST-VNE algorithm. In this section, we aim to compare the performance of SAST-VNE to the following well-known VNE baselines:

- CEVNE (65);
- VINEYard (101);
- SN-VNE (102).

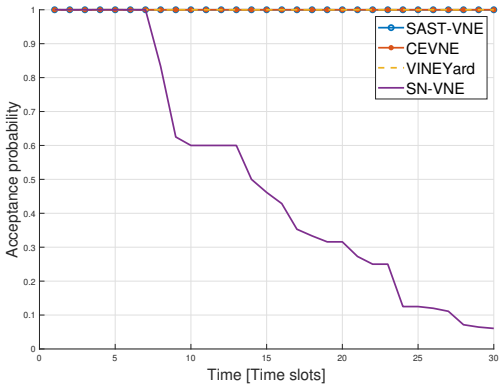
The first two baselines run a joint optimization of the node and link mapping, as summarized in Table 5.3, with the construction of the augmented substrate graph. SN-VNE, instead, is a satellite-based VNE algorithm, which deals with the dynamic nature of the substrate network in an heuristic manner, using a shortest-path based approach.

Table 5.3: Baselines' features comparison

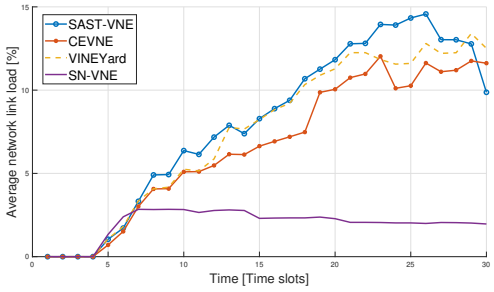
Algorithm	Network	QoS Policy	Optimization
CEVNE (65)	static graph	best-effort	load-balancing and congestion-aware
VINEYard (101)	static graph	best-effort	load-balancing
SN-VNE (102)	satellite network	best-effort	shortest path
SAST-VNE	terrestrial-satellite	priority-based	load-balancing and migration minimization

5.7.1 Acceptance probability - Network usage

In this section, our objective is to compare (1) the average acceptance probability per slice over the duration of the simulation and (2) the average link load. The acceptance probability is computed every time slot as the number of mapped slices over the arrived ones. The average load on the link is computed as the average capacity consumption of each substrate link, i.e., defined in percentage. For this performance, we do not differentiate the class of the slices. Intuitively, the better the load-balancing, the higher the probability of a slice being

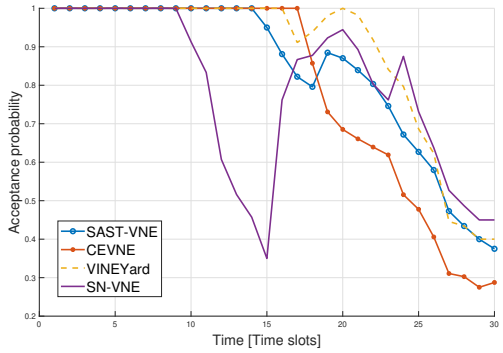


(a) Acceptance probability vs time

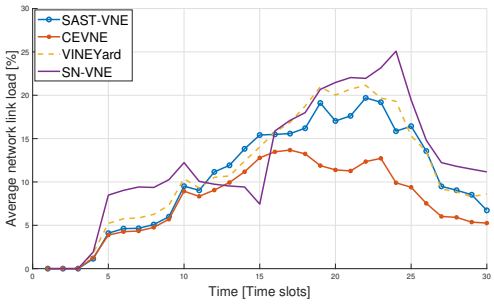


(b) Average link load over time

Figure 5.6: Acceptance probability and average load vs time in low load conditions



(a) Acceptance probability vs time



(b) Average link load over time

Figure 5.7: Acceptance probability and average load vs time in high load conditions

accepted. It is worth mentioning that this is the most tricky and challenging performance to evaluate SAST-VNE with, because it is not implemented to further reduce the load-balancing provided by the baselines and CEVNE, which is already near optimal (due to relaxation), but to minimize the average node and link migrations of each slice to be migrated.

We run two simulations with different level of traffic, i.e., in case of low and high load. Figures 5.6a and 5.6b depict the results for low load. The aim is to show that when the load is low, our algorithm does not worsen the near-optimal solution provided by CEVNE and VINEYard. Figure 5.6a shows the acceptance probability vs. time. SAST-VNE matches VINEYard and CEVNE while SN-VNE, as a heuristic solution, decreases the acceptance probability very quickly. As the network load is low, the near-optimal solutions manage to map all the incoming slices, so the acceptance probability is fixed to 1. On the other hand, the more time passes, the higher the load on the network as the number of mapped slices,

which consume resources, increases. Even in this case, Figure 5.6b shows that SAST-VNE does not deviate too much from the near-optimal solutions.

Similarly, we analyze the comparison under high load conditions (see Figure 5.7). Even in this case, we see that SAST-VNE behaves similarly to CEVNE and in both figures. SN-VNE has a "wave" behavior in Figure 5.7a because with the higher number of arrived slices, the probability that there is one which can be embedded is higher with respect to the low load scenario. This is also reflected in the average link load (Figure 5.7b). It is worth to underline that we do not assume that slices expire, hence, network resources are consuming with the time to avoid the comparison to be influenced by that.

5.7.2 Handover procedures

This section focuses on the analysis of the number of migrations during handovers, due to loss of LoS within a satellite communication, which require parts or the full slice to be migrated towards different areas of the substrate network. It is worth mentioning that handovers are procedures usually known a priori because the trajectories of satellites, and consequently their link connectivity, are pre-established. This is the reason why a relatively high time complexity should not be immediately categorized as a drawback, considering that the computation can be done in a proactive manner. In addition, handovers are unavoidable for Non-Geostationary (NGSO) links but the migration cost should be considered, especially in those cases where only part of the slice is affected.

In fact, in this section, we compare SAST-VNE with the baseline algorithms. We compare the average number of differences in terms of node and link mapping between consecutive time slots when a handover occurred. It is worth underlining that this metric plays a key role because, for any handover (link or node), there is a cost. To simplify the analysis, we do not consider the specific cost, so we focus only on the pure percentage of differences, which is directly proportional to the cost. To measure the efficiency of an algorithm in minimizing node and link handovers, we look at the percentage of nodes and links that are different between consecutive handovers. Then we average over the mapped slices. Clearly, we do not count the slices that are not subjected to handover procedures, as they will affect the average. For this comparison, we subdivide the results per class of slices. We run this simulation with low load to avoid that eventual failures would reduce the number of slices, i.e., samples, from any algorithm and, consequently, would negatively impact the final comparison.

Table 5.4: Average Node migrations during handovers

Slice's class	SAST-VNE	VINEYard	CEVNE	SN-VNE
1	30.2	68.75	65.4	100
2	29.72	72.50	55.36	100
3	15.56	76.1	55.83	99.6

Table 5.5: Average Link migrations during handovers

Slice's class	SAST-VNE	VINEYard	CEVNE	SN-VNE
1	75	100	94.8	100
2	76.67	90.83	95.24	100
3	89.17	92.86	96.67	100

Table 5.4 compares the average migration of the nodes. It can be seen that the percentage of node variations in SAST-VNE is considerably lower with respect to the other algorithms. SN-VNE is a shortest-path based algorithm, so the likelihood that all nodes are different is considerably high. CEVNE and VINEYard perform roughly similarly. This is due to the fact that their only difference is that CEVNE only uses 95% of the available capacity at each link. However, considering that the traffic load is low for this simulation, the probability that a link cannot accommodate the slice anymore is very low. Thus, the overall difference of performance between CEVNE and VINEYard is very limited. Following the same approach, we analyze and compare the average link differences, as shown in Table 5.5. Clearly, the trend matches the previous discussion related to nodes, as they are strictly related. However, for the link comparison, higher values can be observed since, on average, the number of links impacted during handovers is higher than the number of nodes.

5.7.3 Congestion intensity

In this section, we simulate a congestion over some links of the substrate network. To be more precise, for the sake of simplicity, we only simulate sudden congestion on some substrate links. Eventually, congestion can also occur on substrate nodes, such as over-utilization of computability or storage resources, but this is outside the scope of this article. A random algorithm selects a set of links and it reduces their available capacity for a period of time. To avoid simulating congestion in links that are not accommodating links, we select a random number of slices and simulate congestion to a randomly selected number θ of links which accommodate them. Congestion is simulated as 85% of their capacity being used. As previously

mentioned, a relatively high number is selected to make sure that the congestion triggers re-computation of the embedding for that slice. We use $\theta = 30\%$ because higher values are likely to create unavailable paths, especially for the mapped slices over satellite links. First, it is worth mentioning that the more congested links, the higher the probability that more slices are impacted. At the same time, among the impacted slices, some high priority slices may not afford outage time. In general, as we assume that congestions are not easily predictable, a reactive approach is required. Thus, the time complexity of finding a new embedding and the delay or service outage play a relevant role.

With these premises, we introduce the most important novelty developed in our framework. SAST-VNE is flexible in managing congestion depending on the priority of the impacted slice. In fact, if the slice has the highest priority, we assume that the outage must be as smallest as possible, thus, an heuristic VNE algorithm is implemented to minimize as more as possible the embedding time, at the expense of the load-balancing performance. On the other hand, if the impacted slice does not have strict latency requirements, SAST-VNE solves the joint optimization in (5.1), s.t. (5.2)-(5.10), (5.12). In this simulation, we compare the average embedding time and migration cost when a congestion occurs. Even in this analysis we assume low load of traffic to avoid that failures, due to lack of capacity, will impact the comparison between SAST-VNE and the baselines. In Figure 5.8, we compare the average embedding time for the 4 algorithms. Each line represents the average embedding time among the slices with the same priority. For the high priority slices, the difference of SAST-VNE, compared to CEVNE and VINEYard, can be appreciated. In fact, SAST-VNE is more comparable, for high priority slices, to the SN-VNE, as they are both shortest-path based approaches. Differently works for the medium and low priority slices, where we assume low latency constraints are not applicable. In this case, our framework SAST-VNE prioritizes the reduction of cost migration over time complexity. For this reason, SAST-VNE has the highest time complexity compared to the other three baselines. The time complexity of SAST-VNE is always higher than CEVNE and VINEYard due to the added element of the objective function that seeks the minimum migration cost that, as explained in Section 5.5.2, is not a negligible factor. On average, it showed an additional $5 - 10$ s.

In Figure 5.9, on the other hand, we compare the average link difference for each class of slices and for each algorithm. Since we showed in section 5.7.2 that there is a correlation between the average node and link differences, for the sake of simplicity, in this section we

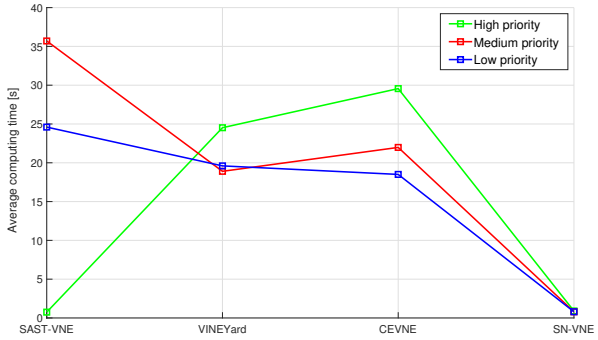


Figure 5.8: Average computing time for different priority

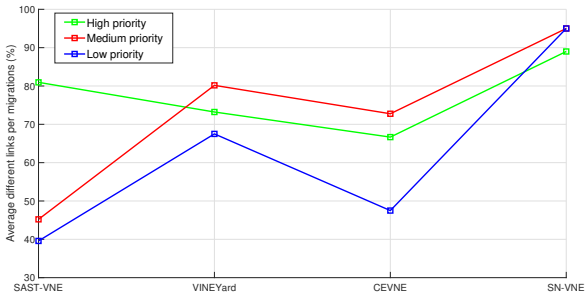


Figure 5.9: Average link migrations for different priority

only consider the average link differences. It can be noted that, even in this case, for the high priority slices, SAST-VNE behaves similarly to the heuristic SN-VNE. In fact, both provide $\sim 10-15\%$ higher migrated links than the other two. Similarly to the previous discussion, for the medium and low priority slices, SAST-VNE improves the three baselines between 8 and 50%.

5.8 Summary

In this chapter, we proposed a novel flexible framework, named SAST-VNE, to implement VNE for network slicing in a dynamic scenario, such as the Beyond 5G integrated satellite-terrestrial network, with a flexible approach. Unlike traditional VNE approaches, which use a fixed mapping scheme for any VNR or slice, we provide a full framework that runs different types of VNE algorithms depending on the priority of the slice being mapped. A joint objective function is provided to maintain load balance while minimizing migration cost. First, as 6G networks will also include NGSO links, SAST-VNE is programmed to handle procedures

such as handovers, due to the satellite movement, in a proactive way, as the satellite movements are known. According to some 6G use cases, we simulate slices with three different priority classes. We show that SAST-VNE, compared to baselines, reduces the migration cost, both for links and nodes, during satellite handovers while minimizing the difference between the previous and current embedding. In addition, we compared the acceptance probability during both low and high load conditions and SAST-VNE proved to match the near-optimal solutions provided by VINEYard and CEVNE. Furthermore, we tested SAST-VNE while randomly generating congestion over some network links. SAST-VNE proved to improve the migration cost for the non-critical slices compared to the baselines. On the contrary, a low-complexity solution is provided for the high priority slice which drastically reduces the computing time and provides an almost immediate solution.

Chapter 6

Discussion and perspectives

This chapter summarizes the main contributions of the thesis and presents some open topics.

6.1 Conclusion

The integration between satellite and terrestrial has paved the way to the development of a connected ecosystem, which provides an ubiquitous, QoS-oriented, and continuous service. On the other side, the complexity of the network increased considerably because (1) satellite and terrestrial links have very different physical properties and stability, (2) the devices involved differentiate considerably in on-board capabilities, and (3) the number of links and interconnections increases, which further complicates the solution of resource allocation algorithms. This thesis had the objective to match one of the most popular resource allocation problems, named VNE, to these novel networks, covering the main lacks of the state-of-the-art in this field. Above all, before describing in detail the main takeaways from the technical chapters, it must be mentioned that the literature has highlighted few contributions where the VNE algorithm was associated to a testbed where real-traffic could be transmitted. Being aware of the added value that evaluation via the testbed could bring, this thesis proposed, in chapter 2, the description of the built-in SDN-based testbed, named MIRSAT, which has hosted the majority of the VNE algorithms presented here.

Chapter 3 focused on providing a VNE solution, named DTA-R (relaxed version due to the MBLP formulation), for an integrated MEO-terrestrial network with the objective of minimizing the traffic migrations, or handovers, during the lifetime of a virtual network request. End-to-end connectivity was provided for a set of incoming VNRs and the results

showed that the a priori knowledge of the dynamic part of the network (satellite-terrestrial links) can be exploited to reduce the traffic migrations. The minimization was executed on a sequential time slots basis, where the differences in the link mapping are minimized. A major trade-off was also discussed. The longer the time view of the objective function, the better the results but at the expense of higher computation.

In chapter 4, a more dynamic solution for VNE was presented. In this scenario, the VNE algorithm is continuously fed with the real-time traffic statistics (throughput) from each mapped VNR. Based on this, the algorithm knows the real-time utilization of each substrate link. As the objective is to increase the acceptance probability of an incoming VNR and to reduce the average substrate capacity consumption, an optimistic approach is used to assign the resources to each VNR, based on the assumption that often services in the real scenario do not need all the required resources. The results have shown the improvement of the acceptance probability, with several parameters' impact analyzed such as the collection period time, the priority-dependent system to assign lower resources than required while keeping the SLA under control.

Finally, in chapter 5, the work in the previous two chapters is expanded to a network slicing scenario for an integrated GEO-LEO-terrestrial network. In fact, the VNE algorithm proposed here, named SAST-VNE, intends to (1) adapt to the dynamic scenario of the network and proactively manage routing handovers, (2) keep track of the throughput of each VNR to see if there is any congestion, and (3) solve congestion scenarios with a re-mapping of the affected services, while minimizing the traffic migrations.

6.2 Future work

In this section, we discuss the extensions of the thesis and potential future works.

(a) The problem proposed in chapter 4 highlighted the benefit of being optimistic when assigning network resources while using a probabilistic description of the traffic. One extension of the work is the application of this approach to the scenario presented in chapter 5, to see how it matches with the high dynamicity of the network;

(b) The presented VNE solutions were mainly oriented toward the routing and, therefore, the link-related resources such as capacity and delay. It is advised, as further work,

that more node-related capabilities, which for instance consider the on-board limitated resources of satellites, are added to the proposed problems;

(c) In all three main contributions, for each substrate link there is an initial capacity that is set. During the simulation, the initial capacity is consumed if VNRs are mapped to that link. In contrast, if the link is not used, the capacity remains unchanged. This is a critical aspect as in the reality, especially in satcom, the capacity of a link can vary based on the elevation angle. It would be also interesting to run the simulations including this dynamicity in the substrate link and to manage the handovers in a proactive manner to minimize the traffic loss;

(d) in chapter 5 we proposed the SAST-VNE algorithm without considering its application to the testbed. This was due to the fact that, unlike the first two VNE algorithms, the testbed application adds one layer more of abstraction from an SDN perspective for the following reason. When there is end-to-end connectivity, each VNR is differentiated with one identifier and the routing is managed using only this. If the E2E becomes a slice, this means that the internal routing must be managed by the SDN controller using a second identifier (e.g., MAC/IP/TCP parameters). An update of the testbed with the capabilities of providing 2-layers of differentiation is considered a strong added value;

(e) The MIRSAT testbed has shown that when the routing is computed from the VNE algorithm (link mapping), this can immediately be applied. The link resources such as the capacity and the delay of the links are properly emulated. However, node-related features, such as processing or storage capabilities, are not emulated and this can also be an improvement of the testbed to make it even more realistic.

Chapter 7

Bibliography

Bibliography

[1] Exponential capability growth. exponential potential. [Online]. Available: https://www.ericsson.com/en/blog/2021/10/exponential-capability-growth-exponential-potential

[2] Cisco Annual Internet Report (2018-2023). [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[3] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[4] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, 2005.

[5] M. Chahbar, G. Diaz, A. Dandoush *et al.*, "A Comprehensive Survey on the E2E 5G Network Slicing Model," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 49–62, 2021.

[6] H. Zhang, N. Liu, X. Chu *et al.*, "Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 138–145, 2017.

[7] I. Afolabi, T. Taleb, K. Samdanis *et al.*, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[8] F. Ansah, M. Majumder, H. de Meer, and J. Jasperneite, "Network slicing : An industry perspective," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1367–1370.

[9] 3GPP, "," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 23.501, 10 2017, version 18.2.2. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144

[10] ——, "," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 28.530, 10 2019, version 15.3.0. [Online]. Available: https://www.3gpp.org/ftp/tsg_sa/WG5_TM/TSGS5_128/SA_86/28530-f30.doc

[11] Sigcom: our projects. [Online]. Available: https://www.uni.lu/snt-en/research-groups/sigcom/research/

[12] End-to-end cognitive network slicing and slice management framework in virtualised multi-domain, multi-tenant 5g networks. [Online]. Available: https://slicenet.eu/

[13] A network slice for every service. [Online]. Available: https://cordis.europa.eu/project/id/723172

[14] Network slicing. [Online]. Available: https://5gobservatory.eu/tag/network-slicing/

[15] ew european 5g group will focus on network slicing, industry verticals. [Online]. Available: https://www.sdxcentral.com/articles/news/new-european-5g-group-will-focus-on-network-slicing-industry-verticals/2017/06/

[16] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[17] Openflow switch specification v1.3. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

[18] O. Foundation, "Applying SDN Architecture to 5G Slicing," 2016.

[19] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[20] A. Vanelli-Coralli, A. Guidotti, T. Foggi, G. Colavolpe, and G. Montorsi, "5g and beyond 5g non-terrestrial networks: trends and research challenges," in *2020 IEEE 3rd 5G World Forum (5GWF)*, 2020, pp. 163–169.

[21] 3GPP, ""Study on New Radio (NR) to support non-terrestrial networks"," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 38.811, 10 2017, version 15.2.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3234

[22] ——, ""Study on solutions for NR to support nonterrestrial networks"," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 38.821, 06 2021, version 16.1.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3525

[23] Satellite integration in terrestrial system. [Online]. Available: https://connectivity.esa.int/artes-elements/satellite-integration-terrestrial-system

[24] M. Rost and S. Schmid, "Charting the complexity landscape of virtual network embeddings," in *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, 2018, pp. 1–9.

[25] A. Fischer, J. F. Botero, M. T. Beck *et al.*, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[26] J. Bao, B. Zhao, W. Yu, Z. Feng, C. Wu, and Z. Gong, "Opensan: A software-defined satellite network architecture," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, New York, NY, USA, 2014, p. 347–348.

[27] J. Feng, L. Jiang, Y. Shen, W. Ma, and M. Yin, "A scheme for software defined ors satellite networking," in *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*, 2014, pp. 716–721.

[28] Y. Shi, Y. Cao, J. Liu, and N. Kato, "A cross-domain sdn architecture for multi-layered space-terrestrial integrated networks," *IEEE Network*, vol. 33, no. 1, pp. 29–35, 2019.

[29] B. Yang, Y. Wu, X. Chu, and G. Song, "Seamless handover in software-defined satellite networking," *IEEE Communications Letters*, vol. 20, no. 9, pp. 1768–1771, 2016.

[30] T. Li, H. Zhou, H. Luo, and S. Yu, "Service: A software defined framework for integrated space-terrestrial satellite communication," *IEEE Transactions on Mobile Computing*, vol. 17, no. 3, pp. 703–716, 2018.

[31] B. Feng, H. Zhou, H. Zhang, G. Li, H. Li, S. Yu, and H.-C. Chao, "Hetnet: A flexible architecture for heterogeneous satellite-terrestrial networks," *IEEE Network*, vol. 31, no. 6, pp. 86–92, 2017.

[32] [Online]. Available: https://ryu.readthedocs.io/en/latest/getting_started.html

[33] [Online]. Available: http://mininet.org/

[34] OpenVSwitch main page. [Online]. Available: https://www.openvswitch.org/

[35] Matlab main page. [Online]. Available: https://nl.mathworks.com/products/matlab.html

[36] [Online]. Available: https://ostinato.org/

[37] [Online]. Available: https://www.agi.com/products/stk

[38] [Online]. Available: https://www.ses.com/o3b-mpower

[39] CpQD switch main page. [Online]. Available: https://github.com/CPqD/ofsoftswitch13

[40] Ostinato api architecutre. [Online]. Available: https://userguide.ostinato.org/architecture

[41] L. Lei, Y. Yuan, T. X. Vu *et al.*, "Dynamic-Adaptive AI Solutions for Network Slicing Management in Satellite-Integrated B5G Systems," *IEEE Network Magazine*, 2021.

[42] O. Kodheli, E. Lagunas *et al.*, "Satellite Communications in the New Space Era: A Survey and Future Challenges," *IEEE Communications Surveys Tutorials*, vol. 23, no. 1, pp. 70–109, 2021.

[43] H. Al-Hraishawi, S. Chatzinotas, and B. Ottersten, "Broadband Non-Geostationary Satellite Communication Systems: Research Challenges and Key Opportunities," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6.

[44] M. Rost and S. Schmid, "On the Hardness and Inapproximability of Virtual Network Embeddings," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 791–803, 2020.

[45] H. Cao, S. Wu, Y. Hu *et al.*, "A survey of embedding algorithm for virtual network embedding," *China Communications*, vol. 16, no. 12, pp. 1–33, 2019.

[46] A. Hashmi and C. Gupta, "A Detailed survey on Virtual Network Embedding," in *2019 International Conference on Communication and Electronics Systems (ICCES)*, 2019, pp. 730–737.

[47] G. Sun, H. Yu, L. Li, V. Anand, Y. Cai, and H. Di, "Exploring online virtual networks mapping with stochastic bandwidth demand in multi-datacenter," in *Photonic Network Communications*, 2012.

[48] H. Di, H. Yu, V. Anand, L. Li, G. Sun, and B. Dong, "Efficient online virtual network mapping using resource evaluation," in *Journal of Network and Systems Management*, vol. 20, 2012, pp. 468–488.

[49] A. Blenk, P. Kalmbach, P. van der Smagt, and W. Kellerer, "Boost online virtual network embedding: Using neural networks for admission control," in *2016 12th International Conference on Network and Service Management (CNSM)*, 2016, pp. 10–18.

[50] K. T. Nguyen and C. Huang, "An Intelligent Parallel Algorithm for Online Virtual Network Embedding," in *2019 International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2019, pp. 1–5.

[51] C. Aguilar-Fuster, M. Zangiabady, J. Zapata-Lara, and J. Rubio-Loyola, "Online Virtual Network Embedding Based on Virtual Links' Rate Requirements," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1630–1644, 2018.

[52] M. Zangiabady, C. Aguilar-Fuster, and J. Rubio-Loyola, "A virtual network migration approach and analysis for enhanced online virtual network embedding," in *2016 12th*

International Conference on Network and Service Management (CNSM), 2016, pp. 324–329.

[53] J. F. Botero, X. Hesselbach, A. Fischer, and H. de Meer, "Optimal mapping of virtual networks with hidden hops," in *Telecommunication Systems*, 2012.

[54] J. Liu, X. He, T. Chen *et al.*, "SN-VNE: A Virtual Network Embedding Algorithm for Satellite Networks," in *2019 IEEE/CIC International Conference on Communications Workshops in China (ICCC Workshops)*, 2019, pp. 1–6.

[55] H. Kuang and Y. Fu, "VNE-TS: A novel virtual network mapping algorithm in SDN," in *2016 5th International Conference on Computer Science and Network Technology (ICCSNT)*, 2016, pp. 657–661.

[56] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, 2009, p. 81–88.

[57] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, 2012.

[58] W. Liu, Y. Xiang, S. Ma, and X. Tang, "Completing virtual network embedding all in one mathematical programming," in *2011 International Conference on Electronics, Communications and Control (ICECC)*, 2011, pp. 183–185.

[59] T. Trinh, H. Esaki, and C. 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.

[60] C. K. Dehury and P. K. Sahoo, "DYVINE: Fitness-Based Dynamic Virtual Network Embedding in Cloud Computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1029–1045, 2019.

[61] Q. Lin, Y. Huang, and Y. Li, "A New Algorithm for Virtual Networks Reconfiguration with Adaptive Interval," in *2018 2nd IEEE Advanced Information Manage-*

ment,Communicates,Electronic and Automation Control Conference (IMCEC), 2018, pp. 2567–2571.

[62] P. Wang, X. Zhang, S. Zhang *et al.*, "Time-Expanded Graph-Based Resource Allocation Over the Satellite Networks," *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 360–363, 2019.

[63] T. Zhang, H. Li, S. Zhang *et al.*, "STAG-Based QoS Support Routing Strategy for Multiple Missions Over the Satellite Networks," *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 6912–6924, 2019.

[64] D. Yang, J. Liu, R. Zhang, and T. Huang, "Multi-Constraint Virtual Network Embedding Algorithm For Satellite Networks," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[65] M. Pham, D. B. Hoang, and Z. Chaczko, "Congestion-Aware and Energy-Aware Virtual Network Embedding," *IEEE/ACM Transactions on Networking*, vol. 28, no. 1, pp. 210–223, 2020.

[66] M. Minardi, S. K. Sharma, S. Chatzinotas, and T. X. Vu, "A Parallel Link Mapping for Virtual Network Embedding with Joint Load-Balancing and Energy-Saving," in *IEEE International Mediterranean Conference on Communications and Networking (MEDITCOM) 2021*, 2021, pp. 1–6.

[67] Q. Lu and C. Huang, "Distributed Parallel VN Embedding Based on Genetic Algorithm," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, 2019, pp. 1–6.

[68] A. Song, W.-N. Chen, T. Gu *et al.*, "Distributed Virtual Network Embedding System With Historical Archives and Set-Based Particle Swarm Optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 2, pp. 927–942, 2021.

[69] [Online]. Available: https://www.ses.com/our-coverage/teleport-map

[70] J. Fraire, P. Madoery, and J. Finochietto, "Traffic-aware contact plan design for disruption-tolerant space sensor networks," *Ad Hoc Networks*, vol. 47, 05 2016.

[71] J. Alonso and K. Fall, "A linear programming formulation of flows over time with piecewise constant capacity and transit times," Tech. Rep., 2003.

[72] [Online]. Available: http://lpsolve.sourceforge.net/5.1/absolute.htm

[73] [Online]. Available: https://www.agi.com/products/stk

[74] [Online]. Available: https://wwwen.uni.lu/layout/set/print/snt/research/sigcom/sdn_lab

[75] F. Mendoza, M. Minardi, S. Chatzinotas *et al.*, "An SDN Based Testbed for Dynamic Network Slicing in Satellite-Terrestrial Networks," in *IEEE International Mediterranean Conference on Communications and Networking (MEDITCOM) 2021*, Athens, Greece, 2021.

[76] Y. Xue, J. Peng, K. Han, and Z. Zhu, "On table resource virtualization and network slicing in programmable data plane," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 319 – 331, March 2020.

[77] H. Cao, Y. Zhu, G. Zheng, and L. Yang, "A novel optimal mapping algorithm with less computational complexity for virtual network embedding," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 356 – 371, Nov. 2017.

[78] I. Maity, T. X. Vu, S. Chatzinotas, and M. Minardi, "D-ViNE: Dynamic Virtual Network Embedding in Non-Terrestrial Networks," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 166–171.

[79] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy, "Adaptive virtual network provisioning," in *Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*. New York, NY, USA: Association for Computing Machinery, 2010, p. 41–48. [Online]. Available: https://doi.org/10.1145/1851399.1851407

[80] G. Sun, H. Yu, V. Anand, and L. Li, "A cost efficient framework and algorithm for embedding dynamic virtual network requests," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1265–1277, 2013, special section: Hybrid Cloud Computing.

[81] M. Lu, Y. Lian, Y. Chen, and M. Li, "Collaborative Dynamic Virtual Network Embedding Algorithm Based on Resource Importance Measures," *IEEE Access*, pp. 1–17, 2018.

[82] D. Chen, X. Qiu, Z. Qu, S. Zhang, and W. Li, "Algorithm for virtual nodes reconfiguration on network virtualization," in *2011 International Conference on Advanced Intelligence and Awareness Internet (AIAI 2011)*, 2011, pp. 333–337.

[83] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, p. 17–29, mar 2008.

[84] N. Farooq Butt, M. Chowdhury, e. M. Boutaba, Raouf', L. M. Feeney, D. Rubenstein, and S. V. Raghavan, "Topology-awareness and reoptimization mechanism for virtual network embedding," in *NETWORKING 2010*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 27–39.

[85] G. Schaffrath, S. Schmid, and A. Feldmann, "Optimizing long-lived cloudnets with migrations," in *2012 IEEE Fifth International Conference on Utility and Cloud Computing*, 2012, pp. 99–106.

[86] K. Qu, W. Zhuang, Q. Ye, X. Shen, X. li, and J. Rao, "Dynamic Flow Migration for Embedded Services in SDN/NFV-Enabled 5G Core Networks," *IEEE Transactions on Communications*, vol. 68, no. 4, pp. 2394–2408, 2020.

[87] Y. Yuan, C. Wang, C. Wang, B. Zhang, S. Zhu, and N. Zhu, " A Novel Algorithm for Embedding Dynamic Virtual Network Request," in *2015 2nd International Conference on Information Science and Controller Engineering*, 2015, pp. 1–5.

[88] S. R. Chowdhury, R. Ahmed, N. Shahriar, A. Khan, R. Boutaba, J. Mitra, and L. Liu, "Revine: Reallocation of virtual network embedding to eliminate substrate bottlenecks," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 116–124.

[89] F. Xu, Y. Li, H. Wang, P. Zhang, and D. Jin, "Understanding mobile traffic patterns of large scale cellular towers in urban environment," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1147–1161, 2017.

[90] L. Yu, H. Shen, Z. Cai, L. Liu, and C. Pu, "Towards bandwidth guarantee for virtual clusters under demand uncertainty in multi-tenant clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, pp. 450–465, 2018.

[91] S. Xu, P. Li, S.-Y. Guo, and X. Qiu, "Fiber-wireless network virtual resource embedding method based on load balancing and priority," *IEEE Access*, vol. 6, pp. 33 201–33 215, 2018.

[92] J. Cai, X. Nian, H. Gu, and L. Zhang, "A user priority-based virtual network embedding model and its implementation," in *2013 IEEE 4th International Conference on Electronics Information and Emergency Communication*, 2013, pp. 33–36.

[93] F. Sadia, N. Jahan, L. Rawshan, M. T. Jeba, and T. Bhuiyan, "A priority based dynamic resource mapping algorithm for load balancing in cloud," in *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, 2017, pp. 176–180.

[94] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[95] A. Vanelli-Coralli, G. E. Corazza, M. Luglio, and S. Cioni, "The isicom architecture," in *2009 International Workshop on Satellite and Space Communications*, 2009, pp. 104–108.

[96] D. Chemodanov, F. Esposito, P. Calyam, and A. Sukhov, "A constrained shortest path scheme for virtual network service management," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 127–142, 2019.

[97] M. Asad, A. Basit, S. Qaisar, and M. Ali, "Beyond 5G: Hybrid End-to-End Quality of Service Provisioning in Heterogeneous IoT Networks," *IEEE Access*, vol. 8, pp. 192 320–192 338, 2020.

[98] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network Slicing in 5G: Survey and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[99] M. Chahbar, G. Diaz, A. Dandoush, C. Cérin, and K. Ghoumid, "A Comprehensive Survey on the E2E 5G Network Slicing Model," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 49–62, 2021.

[100] S. Wijethilaka and M. Liyanage, "Survey on Network Slicing for Internet of Things Realization in 5G Networks," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 957–994, 2021.

[101] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, 2012.

[102] J. Liu, X. He, T. Chen *et al.*, "SN-VNE: A Virtual Network Embedding Algorithm for Satellite Networks," in *2019 IEEE/CIC International Conference on Communications Workshops in China (ICCC Workshops)*, 2019, pp. 1–6.

[103] M. Minardi, T. X. Vu, L. Lei, C. Politis, and S. Chatzinotas, "Virtual Network Embedding for NGSO Systems: Algorithmic Solution and SDN-Testbed Validation," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022.

[104] M. Minardi, Y. Drif, T. Vu, I. Maity, C. Politis, and S. Chatzinotas, "SDN-based Testbed for Emerging Use Cases in Beyond 5G NTN-Terrestrial Networks," in *2nd International Workshop on Autonomous Network Management in 5G and Beyond Systems*, 2023.

[105] Y. Drif, E. Lavinal, E. Chaput, P. Berthou, B. T. Jou, O. Grémillet, and F. Arnal, "Slice Aware Non Terrestrial Networks," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, 2021, pp. 24–31.

[106] T. Ahmed, A. Alleg, R. Ferrus, and R. Riggio, "On-Demand Network Slicing using SDN/NFV-enabled Satellite Ground Segment Systems," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018, pp. 242–246.

[107] S. Hendaoui and C. N. Zangarz, "Leveraging SDN slicing isolation for improved adaptive satellite-5G downlink scheduler," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, 2021, pp. 1–5.

Appendix A

The code of the testbed can be found at https://gitlab.uni.lu/mminardi/sdn-testbed-v.4.