University of Ain Temouchent Belhadj Bouchaib
Faculty of Science and Technology
Department of Mathematics and Computer Science

Resource Optimization in 5G network slicing

Submitted by :

Hamza Ghitri
Aya Boudaoud

Supervised by:

Dr. Ali BENZERBADJ

Graduation Project Report submitted in partial fulfilment
of the requirements of Bachelor's Degree
in: **Computer Science**
Speciality: **Computer Systems**

Academic Year : 2022/2023

Acknowledgement

With hearts full of gratitude, we prostrat ourselves in deepest gratitude to Allah who gave us the courage and determnation to complete this project.

We would like to express our heartfelt gratitude to Mr. Ali Benzerbadj for his invaluable guidance and support throughout this project. His willingness to entrust us with this subject and his ongoing advice proved instrumental in our success. We especially appreciate the time he dedicated to us, which significantly enhanced our understanding and overall performance.

A heartfelt thank you to our family and friends for their support and encouragement to us from near and far.

We are particularly grateful to the jury for their commitment, which played a vital role in this process.

Thank you all for being a part of this important chapter in our academic and personal life..

Contents

List of Figures

List of Tables

Listings

Acronyms

TERM	DESCRIPTION
1G	First Generation
2G	Second Generation
3G	Third Generation
4G	Fourth Generation
5G	Fifth Generation
LTE	Long-Term Evolution
IoT	Internet of Things
eV2X	enhanced vehicle-to-everything
eMBB	Enhanced Mobile Broadband Communications
uRLLC	Ultra Reliable and Low Latency Communications
mMTC	Massive Machine Type Communications
E2E	End-to-End
CriC	Critical Communications
ICT	Information and Communication Technology
MIoT	Massive Internet of things
NFV	Network Function Virtualization
QoS	Quality of Service
SDN	Software Defined Networking
LP	Linear programming
ILP	Integer Linear programming
VM	Virtual Machine

Introduction

Mobile communications have evolved from systems capable of sending voice chats to millions of users to networks supporting trillions of devices. The upgrade to 5G will revolutionize not only mobile devices but also a wide array of consumer products, offering unprecedented speed and service quality, and enabling new use cases, applications, and business models.

With 5G approaching, optical network operators are restructuring their networks to deploy alternative topologies on demand using the same infrastructure. Central to 5G architecture is network slicing, which allows for the creation of logical networks tailored to specific business needs. These slices, spanning the 5G RAN, transport, and core domains, utilize both physical and virtual network resources, which can be dedicated to or shared among slices.

In our research we commence our discussion by exploring 5G technology, highlighting its most significant features that distinguish it from previous generations. Key use cases such as Enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communications (uRLLC), and Massive Machine Type Communications (mMTC) are examined in detail. Following this, we delve into the critical features that facilitate the creation of network slices.

Subsequently, we provide a concise overview of Software-Defined Networking (SDN) architecture, along with the tools that support network slicing. Additionally, we review the Network Functions Virtualization (NFV) framework, emphasizing its role in cost reduction and efficiency enhancement, culminating with an example that demonstrates the implementation of network slicing.

In the second part of our study, we present an overview of Combinatorial Optimization and Integer linear programming. We focused on the concept of fairness and its importance in Network slicing. We also provide an example of a solver used for linear programming problems, detailing its application and usage.

1 5G Technology

1.1 history and definition

The world is changing fast, and the speed of innovation and the adoption of new technologies is increasing at an exponential rate. Although the fourth-generation 4G system was deployed in 2011 and offered broadband mobile services faster than the previous 3G, the race to achieve more advanced mobile traffic is still growing. However, the need for more sophisticated broadband services will push the standards of the current system beyond its limit to provide higher speeds of mobile communications. Nowadays, the evolution of fifth-generation 5G technology will offer more reliable and faster connections on smartphones and other devices. It will provide an internet connection with an approximate speed of one Gigabit per second, which will bring a significant rise in the Internet of Things Technology IoT [1].

1G : The Birth of Mobile Telephony It all begins with 1G, the first generation of mobile networks. Characterized by analog voice calls, 1G laid the foundation for mobile communication, allowing people to make wireless calls for the first time. However, the network infrastructure was basic compared to what we know today and data transmission was nonexistent, limiting its capabilities.[2]

2G : The Era of Digital Communication 2G emerged in the early 1990 marking a significant leap forward by introducing digital communication to mobile networks. This shift from analog to digital paved the way for more efficient voice calls and the introduction of Short Message Service (SMS).

3G : The Dawn of Mobile Internet The advent of 3G technology signaled a major change, offering increased data transfer speeds and paving the way for mobile broadband. Users gained the ability to browse the internet, stream media, and participate in video calls. [3]

4G : The Rise of High-Speed Connectivity The introduction of 4G networks in the 2010s marked a significant advancement in data transfer speeds. 4G

enabled seamless streaming, high-quality video calls, and faster internet browsing. This expansion of 4G networks also fueled the widespread adoption of smartphones, mobile apps, and the mobile internet ecosystem. 4G technology boasted better multimedia services , higher speed and more security that we thought 4G was sufficient and the maximum the domain of wireless technology could ever evolve to. However, the introduction of 5G challenged this perception.

5G : The Technological Revolution As stated previously, the increasing demand for high-quality services and the growing consumption of multimedia services have prompted a significant transformation in network management, focusing on abstraction, segregation, and the mapping of forwarding, control, and management aspects. The key differentiates between 4G and 5G lie in speed, bandwidth, and latency. 5G is anticipated to offer the fastest data rates compared to 4G LTE, with latency under 5 milliseconds. Additionally, while 4G networks primarily focus on user-level security, such as data encryption and network-level security, 5G networks operate at a more sophisticated level, addressing security concerns at the business, delivery model, and service levels. This necessitates multiple layers of security to prevent attacks like Denial of Service.

Softwarization is a key aspect of 5G networks, enabling Network Slicing and virtualization of hardware. With softwarization, 5G networks can offer users higher quality and more cost-effective services. Developers and operators can swiftly create application-aware networks and network-aware applications, optimizing performance by tracking connected applications and adjusting performance based on network conditions.

ICT has long been acknowledged as crucial for societal and economic advancement due to its introduction of new conveniences and benefits.

5G technology stands out as a digital platform offering voice and data capacity, with unique capabilities tailored for IoT (Internet of Things), as well as AR (Augmented Reality) and VR (Virtual Reality) applications [4].

1.2 5G Mobile Services

The domain of 5G services extends beyond personal communication, reaching into different aspects of society, including mobile phones, wearable devices, sensors, actuators, vehicles, robots, and more. Consequently, it emerges as crucial infrastructure, fostering societal innovation alongside advancements in the ICT sector.

This section classifies different 5G services into five groups, focusing on their characteristics as experienced by end-users rather than solely on the underlying technology. Furthermore, it examines scenarios of 5G mobile services and the specific requirements of each service category. Before delving into the scenarios, it is imperative to understand the necessity of 5G mobile services from the perspective of end-users.

As shown in Figure 1, the new service scenarios can be grouped into five categories.

Figure 1: Categorization of 5G mobile service scenarios

1.2.1 Immersive services

- Augmented reality / virtual reality : VR and AR are poised to play crucial roles in future communication, as depicted in Figure 2.

Figure 2: Real-time mixed reality with AR and VR services.

1.2.2 Intelligent services

- Crowded area services : crowded areas demand efficient wireless service. 5G, with small-cell networks and mobile edge computing, eases congestion but poses a cost-quality trade-off. Energy-efficient strategies and front haul optimization are crucial for intelligent services.

Figure 3: Crowded area services.

1.2.3 Omnipresent 5G Services

- Internet of Thing : 5G introduces diverse devices, surpassing 4G's capabilities. Wearables and home devices fuel extensive data exchange, driving IoT growth in personal networks, buildings, and cities. Smart functionalities span health monitoring, security, and traffic management

Figure 4: Internet of things.

1.2.4 Public services

Private Security and Public Safety : In disasters, rapid network reconstruction is critical, using mobile base stations and wireless core networks. Infrastructureless networks with mobile devices provide essential services. Network-connected CCTVs enhance safety monitoring. Drones and robots aid rescuers and relay information. 5G offers advanced security and prioritizes public safety data. PS-enabled terminals enable direct communication and energy-efficient operation.

Figure 5: Private security and public safety services.

1.2.5 Emergency services

In emergency situations, providing patient data in ambulances and remote medical care enhances responsiveness. First-aid robots can deliver medical treatment in remote areas, but reliability and low latency are crucial limitations.

1.3 Characteristics and functionalities of 5G :

1.3.1 Characteristics

- Supports virtual private network.
- Presents a high resolution for sharp, passionate cell phone every day and give consumers well shaped and fast Internet access.
- The uploading and downloading speed of 5G technology touching the peak.
- The 5G technology network offering enhanced and available connectivity just about the world.

1.3.2 functionalities

The major difference, from a user point of view, between current generations and expected 5G techniques must be something else than increased maximum throughput, other requirements include:

- Lower out age probability, better coverage and high data rates available at cell edge.
- Lower battery consumption.
- Wearable devices with AI capabilities.
- Multiple concurrent data transfer paths.
- More secure, better cognitive radio/SDR Security.

- World Wide wireless web (WWWW).
- More applications combined with artificial intelligent (AI) as human life will be surrounded by artificial sensors which could be communicating with mobile phones. Not harmful to human health.
- Cheaper traffic fees due to low infra structure deployment costs [5].

1.4 5G use cases

5G technology is not just about faster internet; it is about unlocking a world of possibilities. From ultra-fast streaming to autonomous vehicles, 5G use cases span a wide spectrum of applications. In this overview, we'll explore the key use case categories each poised to revolutionize industries and reshape the way we live, work, and connect [6].

1.4.1 Enhanced Mobile Broadband Communications

eMBB use cases aim to provide broadband accesses with up to 10Gbps bandwidth to enable different service scenarios, such as dense urban society, i.e., stadium, Ultra-High Definition (UHD) Videos streaming, Cloud Storage, Moving Hot-spots and AR. They facilitate the support for the services with high data rates. Specifically, they can deal with huge data traffic volumes and offer wide area connectivity and coverage.

1.4.2 Ultra Reliable and Low Latency Communications

uRLLC use cases can assure the services with ultra-low latency connectivity, ultra-high reliability and availability, such as interactive tactile Internet, automated traffic control, automatic driving, AR/VR, collaborative robots and remote object manipulation.

1.4.3 Massive Machine Type Communications :

mMTC use cases can facilitate the network connectivity for high density of devices, especially the non-latency sensitive devices, in ultra-dense scenarios with broadband accesses. They can guarantee the high density of network

connectivity for users with various smart and intelligent devices, i.e., Smartphones and Smart Wearables, in the area of Smart Homes/Cities and Smart Farming [7].

1.5 The concept of fairness in 5g network slicing

The widespread adoption of wireless technology has transformed our world by connecting almost everything to the Internet. This shift is evident with the advent of technologies like the Internet of Things (IoT), which rely on the interaction of numerous smart devices to deliver a wide array of intelligent services. In this highly connected era, ensuring fairness among these devices has become a crucial issue that needs to be addressed from multiple perspectives, including energy consumption, quality of service (QoS), spectrum sharing, and more.

Fairness in wireless networks generally pertains to the allocation or sharing of resources. It aims to ensure that resources are distributed equitably, taking into account the diverse needs and expectations of system users.

Unfair resource distribution can lead to resource starvation, where some users or nodes do not receive essential resources, negatively impacting performance and resource efficiency. Fairness research has focused on addressing this issue by developing strategies and protocols that promote equitable resource distribution. [8]

2 Network Slicing

2.1 Definition

A network slice is an End-to-End (E2E) logical network, running on a shared underlying (physical or virtual) infrastructure, that offers certain network capabilities and network characteristics to fulfill a given business goal of a customer.

When we create a network slice, that slice is logically separated from all other

network slices and will support a set of attributes, qualities, and characteristics.

We can create multiple network slices, but does require very comprehensive slice orchestration. We need to be able to manage the network slices' life cycle management. Instantiate, maintain, and tear them down when they are no longer required. We need to independently monitor them.

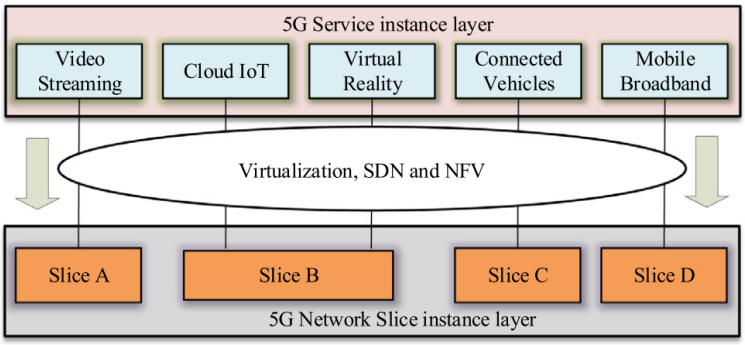

Figure 6: A Framework of the NFV architecture .

2.2 Network slicing use cases

SDN controller	Contribution/Objectives/Functionality
eMBB	to provide high data rates on 5G systems so as to cope with the huge data traffic volumes and UE connectivity per area
CriC	to facilitate mission critical services such as the tactile internet, public safety, emergency response , VR and AR.
eV2X	safty-related services such as remote driving and vehicle platooning
MIoT	provides a common communication connectivity and internetworking for various smart devices in the area of smart cities, homes and farming

3 Network Function Virtualization

3.1 Definition

Telecommunication networks comprise a vast and growing array of proprietary equipment due to the exponential increase in service requests. This growth has led to higher costs and greater complexity in network management. Network Functions Virtualization (NFV) provides a solution by enabling the dynamic allocation of Virtual Network Functions (VNFs), thereby offering flexible network service provisioning and reducing both capital and operating costs [9].

NFV represents a shift towards implementing dedicated functions, traditionally executed in hardware such as routers, firewalls, and load balancers, through software. In this model, NFVs operate within virtual machines, utilizing the processor, memory, and storage resources of the underlying hardware as if they were its own.

By decoupling software from hardware, NFV allows for greater flexibility in deploying network services, as software components are not bound to specific hardware configurations and can perform various tasks independently.[10]

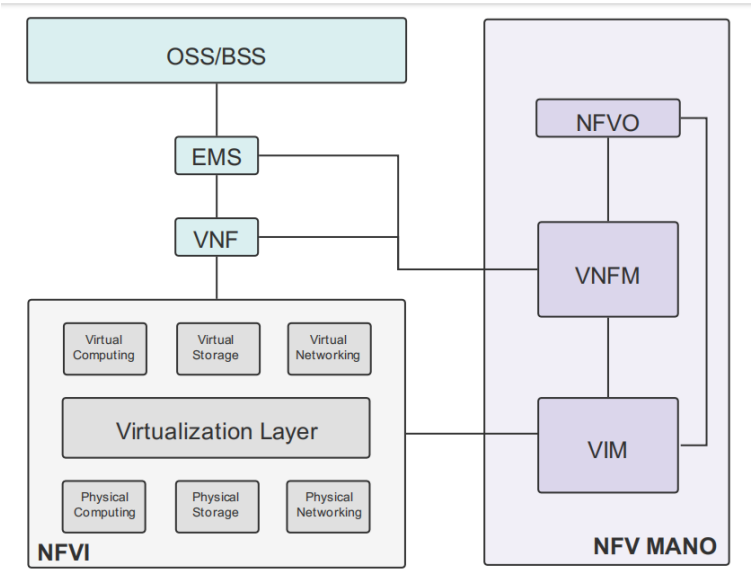

Figure 7: A Framework of the NFV architecture .

3.2 Implementing Network Function Virtualization tools

Implementing Network Function Virtualization (NFV) typically involves a combination of tools and technologies to virtualize network functions and manage them efficiently. Some of the common tools:

3.2.1 Virtualization Platforms

Like hypervisors which are software that enables multiple virtual machines (VMs) to run on a single physical host.
examples : VMware vSphere, Microsoft Hyper-V, and KVM (Kernel-based Virtual Machine).

3.2.2 Orchestration and Management

Like NFV Orchestrators (NFVO) which are Software platforms that automate the deployment, management, and scaling of virtualized network functions.
examples : Open Source MANO (OSM), ONAP (Open Network Automation Platform), and VMware vCloud NFV.

3.2.3 Software-Defined Networking (SDN)

- SDN Controllers: Centralized software platforms that manage the forwarding plane of network devices to enable dynamic, programmable network configurations. OpenDaylight and ONOS are popular open-source SDN controllers.
- SDN Switches: Network switches that can be controlled and programmed by SDN controllers. Examples include Open vSwitch (OVS) and Cisco Nexus switches with SDN capabilities.

4 Software Defined Networking

4.1 Definition

SDN refers to a network architecture where the forwarding state in the data plane is managed by the remote control plane. In another concept, SDN is an approach to plan, deploy, and administer networks that separate the control and data/forwarding planes. SDN creates a dynamic and flexible network architecture that can change as business needs change [6][11].

The architecture is appropriate to support 5G network slicing and to provide the important characteristics necessary for implementing it [12]. The illustration of the architecture of SDN is shown in Figure 13. Additionally, the main SDN architectural components are logically centralized controllers, which can manage network slices effectively and dynamically based on the key principles of network slicing.

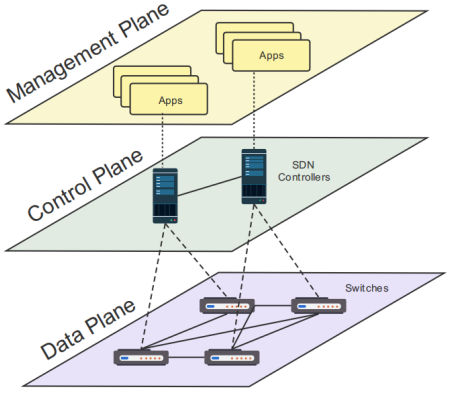

Fig. 2.3 Framework of the SDN architecture.

Figure 8: Framework of the SDN architecture

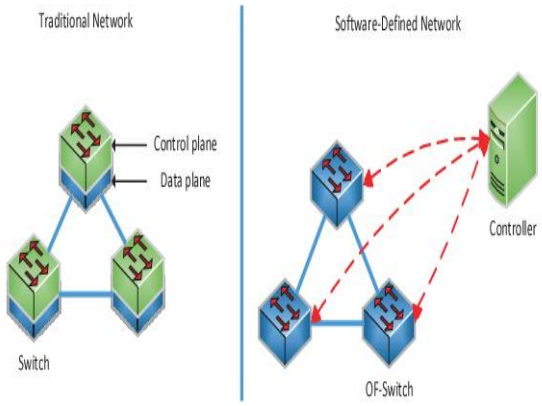

Figure 9: Traditional networks Vs. SDN

4.2 Mininet to implement SDN

Mininet is a popular open source network emulator capable of creating networks of virtual hosts, switches, and SDN controllers .

Mininet was created in Python and provides a Python API for user cus-

tomization -though it does rely on some C programming utilities
It can run in a VM based on Linux Ubuntu server, and using process-based virtualization it can create hundreds or thousands of virtual host and network instances on a single operating system. The Mininet hosts run standard Linux operating systems, and Mininet switches running Linux can support OpenFlow
.
There are many reasons that trigger the use of this emulator. First, there are only few network devices available for the purpose of implementing SDN standard as it is yet not widespread technology from the industrial perspective. In addition, implementing network with large number of network devices is very difficult and costly. Therefore, to overcome these problems, virtual mode strategy has been conducted for the purpose of prototyping and emulating these kind of network technologies.

Creating a Network using Mininet

To create a network using Mininet, there are two ways, a mininet network can be created with one single command as we can typically use Python scripts with the Mininet API.

- One command creation

we can use the mn command-line utility provided .

- Minimal topo with two hosts and one switch : *$ sudo mn*
- Linear topology of k switches, one host per switch :
  $ sudo mn –topo linear,k
- Single switch connected to k hosts :
  $sudo mn –topo single,k
- Single switch connected to k hosts :
  $ sudo mn –topo reersed,k
- Topology for a tree network given depth and fanout :*$ sudo mn –topo tree,n,k*

- using python :

Creating a Mininet topology in Python involves using the Mininet API to define network elements such as switches, hosts, links, and controllers. Here is a step-by-step guide to creating a simple Mininet topology in Python:

1. Import necessary modules from the Mininet library.
2. Define a Python class representing your network topology. You can subclass mininet.topo.Topo to create your custom topology.
3. Inside your topology class, define hosts and switches using the addHost() and addSwitch() methods respectively. You can specify parameters such as host names, IP addresses, and MAC addresses.
4. Connect hosts and switches using the addLink() method. Specify the source and destination nodes, and optionally set link parameters such as bandwidth and delay.
5. Instantiate a Mininet object using your custom topology class.
6. Start the Mininet network using the start() method.
7. you can now interact with the network by running commands, transferring files, or testing network connectivity.
8. After you're done with the network, stop it using the stop() method.

Example :

creating a topology with one switch connected to three hosts :

- using sudo mn command : *$ sudo mn –topo single,3*
- using python :

```python
from mininet.net import Mininet
from mininet.topo import Topo
from mininet.node import OVSSwitch, Host
from mininet.link import TCLink
from mininet.cli import CLI

class MyTopo(Topo):
    def build(self):
        switch = self.addSwitch('s1')
        host1 = self.addHost('h1')
        host2 = self.addHost('h2')
        host3 = self.addHost('h3')

        self.addLink(host1, switch)
        self.addLink(host2, switch)
        self.addLink(host3, switch)

net = Mininet(topo=MyTopo(), switch=OVSSwitch, link=TCLink)

net.start()

CLI(net)

net.stop()
```

Figure 10: Mininet example .

5 Network Slicing Implementation Example

In this section, we will learn how to slice an Openflow network and have each slice controlled by a separate controller. the installation of the software needed is mentioned in the last section "Annex 11".
We will slice a wide-area network (WAN) in two different ways. The WAN shown in the figure11 connects two sites. For simplicity, we will have each site represented by a single OpenFlow switch, s1 and s4, respectively. The sites, s1 and s4, have two paths between them:

- a low bandwidth path via switch s2
- a high bandwidth path via switch s3

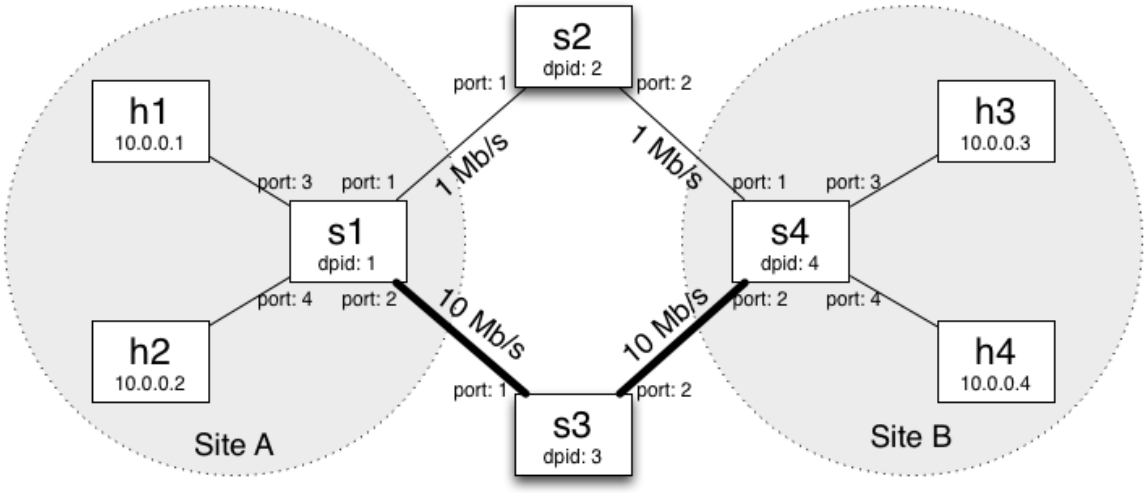

Figure 11: Topology for Flowvisor

5.1 Run the topology

First In a new terminal clear any mininet instance : *$ sudo mn -c* and then start the diamond topology by running the command :

```
\$ sudo mn --custom ~/onstutorial/flowvisor_scripts/flowvisor_topo.py
topo fvtopo --link tc --controller remote --mac --arp
```

This will create a network in mininet with the WAN topology.

After creating the topology using Mininet, we are going to use FlowVisor to enable the network virtualization and slicing. FlowVisor is a network hypervisor that allows the segmentation of a single physical network into multiple, isolated virtual networks, or slices.

5.2 Start Flowvisor

In a new terminal :

- *$ sudo -u flowvisor fvconfig generate /etc/flowvisor/config.json* (Leave the fvadmin password blank by hitting Enter)
- *$ sudo /etc/init.d/flowvisor start* (to start flowvisor)

- *$ fvctl -f /dev/null set-config –enable-topo-ctrl*
- *$ sudo /etc/init.d/flowvisor restart*
- *$ fvctl -f /dev/null get-config* (to ensure that all Openflow switches are connected)
- *$ fvctl -f /dev/null list-slices* (make sure that the only slice is the default fvadmin slice)
- *$ fvctl -f /dev/null list-flowspace* (ensure that there are no existing flowspaces)
- *$ fvctl -f /dev/null list-datapaths*
- *$ fvctl -f /dev/null list-links* (to ensure that all links are up)

5.3 Create slices

- Create a slice named upper connecting to a controller listening on tcp:localhost:10001 and a slice named lower connecting to a controller listening on tcp:localhost:10002

- *$ fvctl -f /dev/null add-slice upper tcp:localhost:10001 admin@upper-slice*
- *$ fvctl -f /dev/null add-slice lower tcp:localhost:10002 admin@lower-slice*
- *$ fvctl f /dev/null addflowspace dpid1port1 1 1 in_port=1 upper=7* (create a flowspace named dpid1port1 with priority value 1 that maps all the traffic on port 1 of switch s1 to the upper slice)
- *$ fvctl f /dev/null addflowspace dpid1port3 1 1 in_port=3 upper=7* (create a flowspace named dpid1port3 that maps all the traffic on port 3 of switch s1 to the upper slice)
- *$ fvctl f /dev/null addflowspace dpid2 2 1 any upper=7*

- *$ fvctl f /dev/null addflowspace dpid2 2 1 any upper=7* (to creat a flowspace for all trafic by using the match value *any*)
- $ fvctl f /dev/null addflowspace dpid4port1 4 1 in_port=1 upper=7
- $ fvctl f /dev/null addflowspace dpid4port3 4 1 in_port=3 upper=7

Now we do the same for the lower slice :

- *$ fvctl f /dev/null addflowspace dpid1port2 1 1 in_port=2 lower=7*
- *$ fvctl f /dev/null addflowspace dpid1port4 1 1 in_port=4 lower=7*
- *$ fvctl f /dev/null addflowspace dpid3 3 1 any lower=7*
- *$ fvctl f /dev/null addflowspace dpid4port2 4 1 in_port=2 lower=7*
- *$ fvctl f /dev/null addflowspace dpid4port4 4 1 in_port=4 lower=7*

Connect each slice to a different instance of the FVexercise controller app. The FVexercise app reactively installs routes based on the destination MAC address, and it is provided as an executable.

Open two new terminals tabs and run the following:

Terminal 1 :

- *$ cd /onstutorial/beaconfvexercisecontroller1*
- *$ ./beacon*

Terminal 2 :

- *$ cd /onstutorial/beaconfvexercisecontroller2*
- *$ ./beacon*

This will launch two instances of FVexercise Beacon controller, listening on port 10001 and 10002 respectively.

The sliced topology should look like :

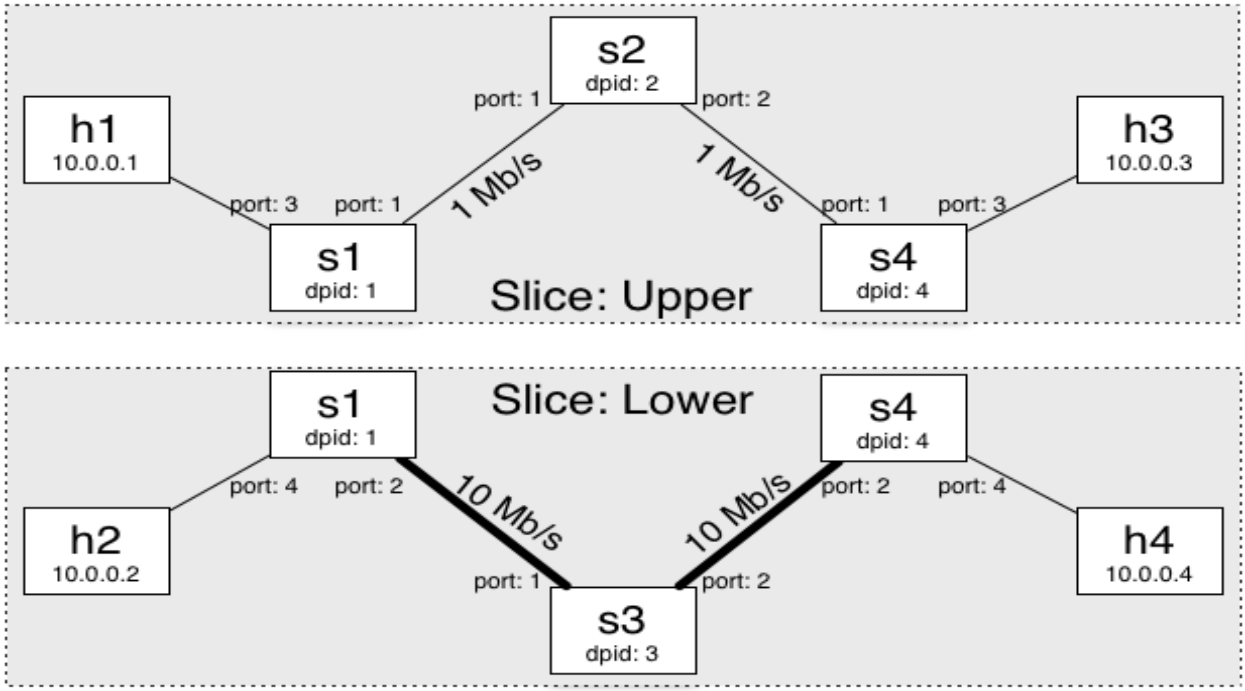

Figure 12: Topology for Flowvisor

to clean and remove the slices that we created w need to run the following commands :

- $ fvctl f /dev/null removeslice upper
- $ fvctl f /dev/null removeslice lower

Source[https://github.com/onstutorial/onstutorial/wiki/Flowvisor-Exercise].

6 Combinatorial Optimization

6.1 Definition

Combinatorial optimization is a subfield of mathematical optimization that is related to operations research, algorithm theory, and computational complexity theory.

Combinatorial optimization refers mainly to the methods employed to solve optimization problems and generally does not offer guidelines on how to convert real-world problems into abstract mathematical questions, and vice versa.

It is considered a solution to some problems such as the traveling salesman problem (tsp) the minimum spanning tree problem (mst) and the knapsack problem. It has important applications in several fields including artificial intelligence, machine learning, auction theory, software engineering, applied mathematics, and theoretical computer science.

Combinatorial optimization is an emerging field at the forefront of combinatorics and theoretical computer science that aims to use combinatorial techniques to solve discrete optimization problems. It seeks to find the optimal solution within a limited set of possible solutions .

During the search process, each solution is evaluated and at the end of the search, the solution with the best value is returned.
The whole field of combinatorial optimization pretty much has its origin in graph theoretic concerns such as edge colorings in undirected graphs and matchings in bipartite graphs [13].

6.2 Linear Programming

Linear programming (LP), also known as linear optimization, is minimizing or maximizing a linear objective function subject to bounds, linear equality, and linear inequality constraints. It serves as a powerful tool for modeling, analyzing, and solving combinatorial optimization problems.

Linear Programming (LP) provides a framework for understanding and approximating combinatorial problems, while insights from combinatorial optimization can enhance LP techniques and formulations. This interplay is central to advancements in optimization theory and practice.

It is used to find the optimal solution of the linear function. In simple term, it is the method to find out how to do something in the best possible way.

6.3 Integer Linear programming

Integer Linear Programming (ILP) involves optimizing a linear objective function subject to a set of linear constraints, where the solution variables must be integers.

This problem is widely applicable in fields such as operations research, logistics, finance, and engineering, where discrete decisions are required [14].

6.4 Solvers

Pulp :

PuLP is an open-source linear programming (LP) library in Python that allows users to describe optimization problems using mathematical notation and solve them using various solvers such as CBC, Cplex, Gurobi. It is particularly useful for problems involving resource allocation, scheduling, and other optimization scenarios. If not specifies, PuLP uses CBC as the default solver.

6.4.1 CBC

Cbc (Coin-or branch and cut) is an open-source mixed integer linear programming problem solver written in C++.MILP combines linear programming (LP) with the added complexity of integer variables . CBC solves these problems using a branch-and-cut algorithm . Can be used as a callable library or using a stand-alone executable. It can be used in a wide variety of ways through various modeling systems, packages, etc.[15]

6.4.2 Xpress

is a commercial, proprietary software that is designed to solve (mixed integer) linear problems [16].Xpress is available on most common computer platforms and also provides several interfaces including a callable library APIs for several programming languages as well as a standalone command-line interface.

6.4.3 CPLEX

CPLEX, standing as an acronym for 'Complex Linear Programming Expert', is a high-performance mathematical programming solver specializing in linear programming (LP), mixed integer programming (MIP), and quadratic programming (QP). It was developed to address the complex optimization needs in various industries, ranging from logistics and finance to manufacturing and energy.[17]

6.4.4 Gurobi

The Gurobi Optimizer is a state-of-the-art solver for mathematical programming. It's solvers were designed from the ground up to exploit modern architectures and multicore processors, using the most advanced implementations of the latest algorithms.It's essentially a solver that helps you find the best possible solution (optimal solution) based on a set of criteria Solves mathematical programming problems:

- Mixed-integer linear programming (MILP) solver
- Mixed-integer quadratic programming (MIQP) solver
- Quadratic programming (QP) solver
- Quadratically constrained programming (QCP) solver
- Mixed-integer quadratically constrained programming (MIQCP) solver

It is proficient in many programming languages, such as Python, Java, C#, MATLAB, etc. Gurobi also proposes a cheerful interactive shell, a graphical interface, and a cloud service for accessibility and ease of deployment.

7 The proposed dynamic resource allocation model

The model that we propos in this report aims to optimize bandwidth allocation in network slicing by maximizing the total allocated bandwidth for all network slices while adhering to link capacity and network slice demand constraints using ILP.

7.1 ILP model description

The proposed ILP model is defined as follows:

$$\max \quad \sum_{i \in S} \sum_{j \in L} \sum_{k \in T} x_{ijk} \tag{1}$$

subject to:

$$\sum_{i \in S} x_{ijk} \leq C[j] \qquad j \in L, k \in T \tag{2}$$

$$\sum_{j \in L} x_{ijk} = D[i][k] \quad i \in S, k \in T \tag{3}$$

$$x_{ijk} \in \{0, 1\} \qquad i \in S, j \in L, k \in T \tag{4}$$

Table 1: List of Sets

Sets	
S	Set of network slices
L	Set of physical links
T	Set of time slots

Table 2: List of Constants

Constants	
D[i][k]	Bandwidth demand of network slice i at time slot k
C[j]	Capacity of physical link j

Table 3: Description of the constraints

(2)	This constraint ensures that the total bandwidth allocated to all slices on a particular link j during time slot k does not exceed the link's capacity C[j].
(3)	This constraint ensures that the allocated bandwidth for each network slice at each time slot k must fulfill the slice's bandwidth demand D[i][k]

Table 4: List of Decision variables

Decision variables	
x_{ijk} $(i \in S, j \in L, k \in T)$	Binary variable indicating whether bandwidth is allocated to network slice i on physical link j during time slot k

7.2 A concrete example

Let us consider an example of 3 network slices (S_1, S_2,S_3), 2 physical links (L_1, L_2) over 2 time slots (T_1, T_2).

1. The bandwidth demands for each network slice at each time slot is as follows:
   - D[S1][T_1]= 5
   - D[S1][T_2]= 3
   - D[S2][T_1]= 3
   - D[S2][T_2]= 1
   - D[S3][T_1]= 1
   - D[S3][T_2]= 3
2. The capacity of each physical link is as follows:
   - C[L_1]= 5
   - C[L_2]= 4
3. The objective function:

   maximize $x_{111} + x_{112} + x_{211} + x_{121} + x_{122} + x_{212} + x_{221} + x_{222} + x_{311} + x_{312} + x_{321} + x_{322}$

4. The constraints:
   Links capcity constraints :

$$x_{111} + x_{211} + x_{311} + x_{112} + x_{212} + x_{312} =< 5$$
$$x_{121} + x_{221} + x_{321} + x_{122} + x_{222} + x_{322} =< 4$$

Demands Constraints :

$$x_{111} + x_{121} = 5$$
$$x_{112} + x_{122} = 3$$
$$x_{211} + x_{221} = 3$$
$$x_{212} + x_{222} = 1$$
$$x_{311} + x_{321} = 1$$
$$x_{312} + x_{322} = 3$$

$$x_{slt} \in \{0, 1\}$$

After we understood the problem that we're going to solve we're going to use AMPL (a powerful tool used to describe and solve complex problems in various fields) to modelize the problem:

1. we write the model file (.mod) that defines the mathematical structure of our problem.
2. create the data file (.dat) that provides specific data values for the parameters defined in our model file.
3. to execute we write the following commands in the CLI.
   - *model <modelFileName>.mod;*
   - *data <dataFileName>.dat;*
   - *option solver NameOfSolver;* (in our case GUROBI)
   - *solve;* (this will show the optimized value of the objective function)
   - *display <variableName>;* (this will show the optimized values of the decision variables

```
set Slices;
set Links;
set Time;


param Demand {Slices, Time};
param Capacity {Links};


var x {Slices, Links, Time} binary;


maximize TotalAllocatedBandwidth: sum {s in Slices
, l in Links, t in Time} x[s, l, t];


subject to Capacity_Constraint {l in Links, t in Time}:
    sum {s in Slices} x[s, l, t] <= Capacity[l];

subject to Demand_Constraint {s in Slices, t in Time}:
    sum {l in Links} x[s, l, t] <= Demand[s, t];
```

Figure 13: DynamicLp.mod

```
set Slices := 'slice 1' 'slice 2' 'slice 3';
set Links := 'link 1' 'link 2';
set Time := '1' '2';


param Demand :=
  'slice 1' '1' 5
  'slice 1' '2' 3
  'slice 2' '1' 3
  'slice 2' '2' 1
  'slice 3' '1' 1
  'slice 3' '2' 3;

param Capacity :=
  'link 1' 5
  'link 2' 4;
```

Figure 14: DynamicLp.dat

we can also use Gurobi solver from python :

Listing 1: Dynamic_Lp.py

```
import gurobipy as gp
from gurobipy import GRB

S = ['slice 1', 'slice 2', 'slice 3']
L = ['link 1', 'link 2']
T = ['1', '2']

D = {
   'slice 1': {'1': 5, '2': 3},
   'slice 2': {'1': 3, '2': 1},
   'slice 3': {'1': 1, '2': 3}
}

C = {'link 1': 5,
     'link 2': 4}

m = gp.Model("MaximizeAllocatedBandwidth")

x = m.addVars(S, L, T, vtype=GRB.BINARY, name="x")

m.setObjective(gp.quicksum(x[s, l, t] for s in S for l in L for t in T),
GRB.MAXIMIZE)

for l in L:
   for t in T:
     m.addConstr(gp.quicksum(x[s, l, t] for s in S) <= C[l],
     f"Capacity_Constraint_{l}_{t}")

for s in S:
```

```
    for t in T:
        m.addConstr(gp.quicksum(x[s, l, t] for l in L) <= D[s][t],
        f"Demand_Constraint_{s}_{t}")

m.optimize()

if m.status == GRB.OPTIMAL:
    for s in S:
        for l in L:
            for t in T:
                print(f"Allocated bandwidth of {s} on {l} at time interval
                {t} ? : {x[s, l, t].X}")
    print("Allocated bandwidth: ", m.objVal, "Mbps")
else:
    print("Model is infeasible or not optimal. Status code:", m.status)
```

8 Conclusion and perspectives

This work allow us to comprehensively understand the transformative potential of 5G technology and its implications for mobile communications. By examining the distinctive features of 5G we highlighted the importance of network slicing in meeting diverce requirements across various domains. We studied the pivotal role of SDN architecture and NFV framework in facilitating the implementation of network slicing. Additionally, we explored the combinatorial optimization and ILP underscoring the importance of fairness in resource allocation, demonstrating the applicability of LP solvers in addressing these challenges.

As a future work, plan to incorporate Artificial Intelligence to dynamically allocate resources for network slicing and also use meta-heuristic methods instead of exact method in order to solve large instances.

9 Abstract

The evolution of mobile communications to 5G promises a transformative shift, accommodating trillions of devices with unprecedented speed and quality. Network operators are adapting by reconfiguring infrastructure for flexible deployment. Central to 5G is network slicing, enabling tailored logical networks for diverse needs across domains like eMBB, uRLLC, and mMTC. This research explores 5G's distinctive features and key use cases, alongside the critical elements enabling network slicing. It delves into SDN architecture and NFV framework, pivotal for slicing implementation, showcasing their role in cost reduction and efficiency enhancement. The study also touches upon Combinatorial Optimization and ILP, emphasizing justice's significance in slicing and providing a solver example for linear programming problems.

Keywords: 5G, network slicing, network virtualisation, software defined networks (SDN), network function virtualization (NFV), resource allocation, Combinatorial Optimization, FlowVisor, Mininet, OpenFlow Protocol

10 Résumé

L'évolution des communications mobiles vers la 5G promet un changement transformateur, pouvant accueillir des milliards d'appareils avec une vitesse et une qualité sans précédent. Les opérateurs de réseaux s'adaptent en reconfigurant l'infrastructure pour un déploiement flexible. Le découpage du réseau est au cœur de la 5G, permettant des réseaux logiques sur mesure pour divers besoins dans des domaines tels que l'eMBB, l'uRLLC et le mMTC. Cette recherche explore les caractéristiques distinctives et les principaux cas d'utilisation de la 5G, ainsi que les éléments critiques permettant le découpage du réseau. Il explore l'architecture SDN et le cadre NFV, essentiels à la mise en œuvre du découpage, démontrant leur rôle dans la réduction des coûts et l'amélioration de l'efficacité. L'étude aborde également l'optimisation combinatoire et l'ILP, soulignant l'importance de la justice dans le découpage et fournissant un exemple de solution pour les problèmes de programmation linéaire.

11 Annex

11.1 Install Mininet (ubuntu 14)

method 1 :

- First you must download Orcale VM virtualBox (or VMware) and ubuntu Workstraion
- Secondly, open the terminal in Ubuntu run these commands :

- *$ sudo apt-get update*
- *$ sudo apt instaall python3-pip*
- *$ sudo pip3 install ryu*
- *$ sudo apt-get install git*
- *$ git clone https://github.com/mininet/mininet* **(needed to get mininet's source code)**
- *$ cd mininet*
- *$ git tag* **(list available versions)**
- *$ git checkout -b mininet-2.3.0 2.3.0* **(or any version greater than 2.3.0)**
- *$ /util/install.sh -a* **(Everything included in the Mininet VM will be installed like Open vSwitch)**

11.2 Install flowvisor

- *$git clone git://github.com/OPENNETWORKINGLAB/flowvisor.git*
- *$ sudo apt-get install ant openjdk-6-jre build-essential* (other versions might work but they weren't tested.)
- *$ cd flowvisor && make*
- *$ sudo make install*
- *$ sudo chown UserName:UserName -R /usr/local/share/db*
- *$ sudo chmod -R 777 /usr/local/share/db*
- *$ rm /usr/local/shar/db/flowvisor/FlowVisorDB/db.lck* (this file prevents flowvisor from running).

Source[https://github.com/syaifulahdan/mininet/blob/master/mininet/flowvisor-instalation.md]

11.3 Install Gurobi

11.3.1 Obtain a Gurobi License

In order to use Gurobi we must first obtain an academic license.
Since our university is not eligible to obtain an academic license, I sent an email to the Gorobi Help Center to request a license and then for verification I was asked mister Riley Clement (Gorobi staff) to send these information :

- academic ID
- school certificate
- Institution website.
- Department name.
- Department website
- Class name

Then a free one year academic license will be created. Provided that this License is used for academic research and/or coursework related to your academic studies.

11.3.2 Install Python

- download python 3.11 (Windows) via Source:[*https://www.python.org/downloads/re 3114/*]
- Add Python to PATH
- Verify the Installation By typing "python –version" in CMD.
  you can check if it was installed successfully by typing the command : *$ python –version*

11.3.3 Download gurobi

Download Gurobi from the Gurobi Downloads page. Determine the appropriate version for my operating system (Windows) And download the installer. Open cmd and run the following command to start the Gurobi license activation process: *grbgetkey your-license-key*
and Install the Gurobi Python interface using pip:

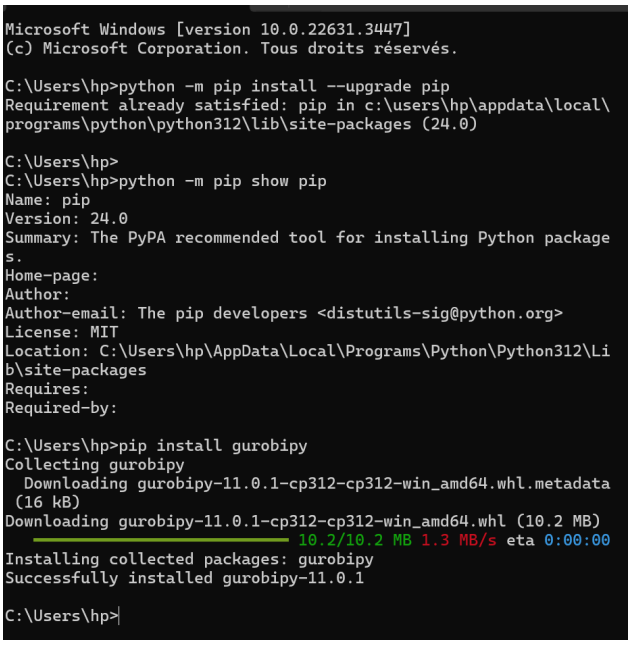

Figure 15: install Gurobi

References

[1] John McCann, Mike Moore, and David Lumb. 5g: everything you need to know. *Tech Radar*, 2019.

[2] Heejung Yu, Howon Lee, and Hongbeom Jeon. What is 5g? emerging 5g mobile services and network requirements. *Sustainability*, 9(10):1848, 2017.

[3] Techeest. From 1g to 5g: The evolution of mobile networks. 2023. Accessed on November 24, 2023.

[4] Akhilesh Kumar Pachauri and Ompal Singh. 5g technology – redefining wireless communication in upcoming years. *International Journal of Computer Science and Management Research*, 1(1), August 2012.

[5] Meenal G Kachhavay and Ajay P Thakare. 5g technology-evolution and revolution. *International Journal of Computer Science and Mobile Computing*, 3(3):1080–1087, 2014.

[6] Ranyin Wang. *End-to-end network slicing design policy in 5G networks*. PhD thesis, King's College London, 2023.

[7] Amal Kammoun, Nabil Tabbane, Gladys Diaz, Abdulhalim Dandoush, and Nadjib Achir. End-to-end efficient heuristic algorithm for 5g network slicing. In *2018 IEEE 32nd international conference on advanced information networking and applications (AINA)*, pages 386–392. IEEE, 2018.

[8] Author Name. *Dissertation in 5G Network Slicing*. PhD thesis, University of Coimbra, Year of Publication. Accessed: Date of Access.

[9] Juliver Gil Herrera and Juan Felipe Botero. Resource allocation in nfv: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, 2016.

[10] Hassan Hawilo, Abdallah Shami, Maysam Mirahmadi, and Rasool Asal. Nfv: State

of the art, challenges, and implementation in next generation mobile networks (vepc). *IEEE Network*, 28(6):18–26, 2014.

[11] Miguel Alves Pimenta. 5g network slicing. Master's thesis, 2023.

[12] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2014.

[13] RealizeTheTerms. Combinatorial optimization, 2020.

[14] GeeksforGeeks. Linear programming, 2024.

[15] J. J. Forrest and R. Lougee-Heimer. Cbc: An open-source mixed integer programming solver. *COIN-OR*, 2007.

[16] Bernhard Meindl and Matthias Templ. Analysis of commercial and free and open source solvers for linear optimization problems. *Eurostat and Statistics Netherlands within the project ESSnet on common tools and harmonised methodology for SDC in the ESS*, 20:64, 2012.

[17] Kenneth Holmström, Anders O Göran, and Marcus M Edvall. User's guide for tomlab/cplex v12. 1. *Tomlab Optim. Retrieved*, 1:2017, 2009.