Network Slicing: Resource Allocation Simulation

Joe Khawand

March 2023

1 Introduction

In the age of the Internet of Things (IoT) and 5G, the demand for network services has increased dramatically, making it more challenging for network operators to provide consistent quality of service (QoS) to all users. Network slicing is a concept in telecommunications that provides a solution to this problem. It enables operators to partition a single physical network into multiple virtual networks, each with its own unique set of network resources, services, and QoS characteristics, tailored to meet the specific requirements of individual users or applications. In this paper, we provide an overview of the concept of network slicing, its benefits, and its applications, and present a simulated example.

2 What is Network Slicing ?

Network slicing [5] is a technique that allows for the creation of multiple virtual networks over a common physical infrastructure. Each virtual network, or slice, is tailored to meet the specific requirements of a particular set of users or applications. These requirements may include QoS parameters such as bandwidth, latency, security, and reliability. The physical infrastructure is partitioned into multiple slices, and each slice can be configured with different network functions and resources, customized to meet the demands of its intended users.

The concept of network slicing is based on the idea that different applications have different requirements for network resources and QoS. By creating dedicated slices for specific applications, operators can provide a customized experience for each user, resulting in higher customer satisfaction.

2.1 Software Defined Networking and Network Function Virtualization in Network Slicing

Network slicing is typically achieved through the use of two related technologies: network function virtualization (NFV) and software-defined networking (SDN). NFV enables the creation of virtual network functions (VNFs), which can be deployed on standard servers rather than dedicated hardware. SDN separates the control plane from the data plane, allowing network administrators to centrally manage and control network traffic.

2.1.1 SDN

Software-defined networking (SDN) is a network architecture that separates the control plane from the forwarding plane. In traditional network architectures, the control plane and forwarding plane are tightly coupled, meaning that network devices (e.g., routers, switches) are responsible for both making forwarding decisions and implementing those decisions. This makes it difficult to program the network, as changes to the network's behavior require making modifications to individual network devices.

In contrast, SDN separates the control plane from the forwarding plane, which enables network administrators to manage the network in a more centralized and programmable way. In SDN, the control plane is responsible for making high-level decisions about how network traffic should be handled and configuring the forwarding plane accordingly. The forwarding plane is responsible for actually forwarding packets through the network based on the instructions received from the control plane.

One of the main benefits of SDN is that it enables network administrators to program the network in a more flexible and dynamic way. For example, they can define network policies and rules in a central controller and then distribute those policies to individual network devices. This makes it easier to manage and optimize network traffic flows, as administrators can modify the network's behavior in real-time in response to changing traffic patterns.

There are several key components of an SDN architecture, including the SDN controller, network applications, and network devices (e.g., switches, routers). The SDN controller is responsible for managing the network's behavior by communicating with network devices using protocols like OpenFlow. Network applications can run on top of the controller to implement specific network policies or services. Network devices are responsible for forwarding packets through the network based on the rules and policies defined by the control plane.

Overall, SDN provides a powerful tool for network administrators to manage and optimize network traffic flows in a centralized and programmable way. By separating the control plane from the forwarding plane, SDN enables network administrators to manage the network more effectively and respond quickly to changing traffic patterns.

It is important to note that this technology has also some drawbacks. The use of SDN technology introduces a single point of failure to the system, which can be a significant vulnerability if the SDN controller breaks down. Depending on the application, an operator may want to ensure a certain level of robustness of the network that may not be achievable with SDN.

2.1.2 NFV

Network Function Virtualization (NFV) is a concept in the networking industry that involves using virtualization technologies to replace traditional network devices or functions with software-based equivalents. In simpler terms, it involves taking the functionality of various networking components such as routers, firewalls, and load balancers, and running them on standard servers, storage, and switches as virtual machines or containers.

NFV provides several benefits to network operators, such as lower capital and operational costs, faster service deployment, increased scalability, and more flexibility to adapt to changing customer needs. By virtualizing network functions, operators can reduce their dependence on proprietary hardware and software and take advantage of the benefits of commodity hardware, automation, and programmability.

Some examples of NFV applications include:

1. Virtualized routers: Traditional routers are expensive, complex, and require specialized hardware. With NFV, routers can be virtualized and run on standard servers, enabling operators to scale their network capacity more efficiently and reduce their overall costs.

2. Virtualized firewalls: Firewalls are critical for network security, but they can also be expensive and complex to manage. With NFV, operators can deploy virtualized firewalls that can be easily scaled and managed using automation tools.

3. Virtualized load balancers: Load balancers are essential for distributing network traffic across multiple servers, but they can also be expensive and difficult to configure. With NFV, operators can deploy virtualized load balancers that can be quickly configured and scaled to meet changing traffic demands.

4. Virtualized network security functions: NFV can also be used to virtualize various network security functions such as intrusion detection and prevention, content filtering, and encryption/decryption, allowing operators to improve network security while reducing costs.

5. Virtualized WAN optimization: WAN optimization involves optimizing network performance across wide-area networks. With NFV, operators can virtualize WAN optimization functions such as compression and deduplication, which can significantly improve network performance and reduce bandwidth costs.

Overall, NFV is a powerful technology that allows operators to transform their networks by replacing traditional hardware-based components with software-based equivalents. The result is a more agile, scalable, and cost-effective network infrastructure that can keep up with the demands of today's digital economy. But this technology comes of course with its drawbacks. Due to virtualization the performance achieved on the machines will always be less than the performance of dedicated hardware. Thus the operators need to chose between performance and flexibility, depending on their applications.

2.1.3 Application to network slicing

In the context of network slicing, NFV and SDN are used to create and manage virtual networks. The virtual networks are configured with customized VNFs, which are deployed on-demand and can be scaled up or down as needed. The SDN controller is responsible for managing the resources of each slice, ensuring that each slice receives the appropriate level of service.

To implement network slicing using NFV and SDN, several challenges must be addressed, such as resource allocation, network slicing orchestration, and service assurance. Resource allocation involves allocating physical resources such as CPU, memory, and storage, to the virtual network functions. Network slicing orchestration involves the creation, instantiation, and management of virtual network functions and slices. Service assurance involves monitoring the network slices for performance issues and taking corrective action if necessary.

Several research studies have proposed solutions to these challenges, such as using machine learning algorithms[4] to optimize resource allocation, using blockchain technology for network slicing orchestration, and using network slicing testing frameworks for service assurance.

2.2 Network Slicing Architecture

The architecture of network slicing is based on the principles of network function virtualization (NFV) and software-defined networking (SDN). In this architecture, the physical infrastructure is divided into multiple virtual networks, each with its own set of network functions and resources. The virtual networks are created and managed by the network slicing orchestrator, which is responsible for allocating the necessary resources, deploying the virtual network functions (VNFs), and ensuring service assurance.

The network slicing architecture consists of several components, including:

1. **Infrastructure:** The physical infrastructure is the underlying network that provides the resources for the virtual networks. The physical infrastructure can be composed of various network technologies, such as fiber optics, wireless, or satellite.

2. **Network Slicing Orchestrator:** The network slicing orchestrator is responsible for creating and managing the virtual networks. It receives requests from the service provider or network operator, and based on the requirements, it creates the appropriate virtual network. The orchestrator is responsible for allocating the necessary resources, deploying the VNFs, and ensuring service assurance.

3. **Virtual Network Function (VNF):** A VNF is a software-based network function that can be deployed on a virtual machine. VNFs can be customized to meet the requirements of a particular network slice. For example, a VNF for a high-bandwidth network slice may have a larger memory allocation than a VNF for a low-bandwidth network slice.

4. **Service Level Agreement (SLA):** An SLA is an agreement between the service provider and the user that specifies the quality of service (QoS) that will be provided. The SLA defines the performance requirements of the network slice, such as bandwidth, latency, and reliability. The SLA is used by the network slicing orchestrator to allocate the necessary resources and ensure that the network slice meets the user's requirements.

5. **Service Assurance:** Service assurance is the process of ensuring that the network slice is meeting the user's requirements. This includes monitoring the network slice for performance issues, such as latency or packet loss, and taking corrective action if necessary. Service assurance is a critical component of network slicing, as it ensures that the user's requirements are being met.

Each of these components plays a critical role in the network slicing architecture. The infrastructure provides the physical resources for the virtual networks, while the network slicing orchestrator creates and manages the virtual networks. The VNFs are customized to meet the requirements of a particular network slice, and the SLA ensures that the network slice is meeting the user's requirements. Finally, service assurance ensures that the network slice is performing as expected.

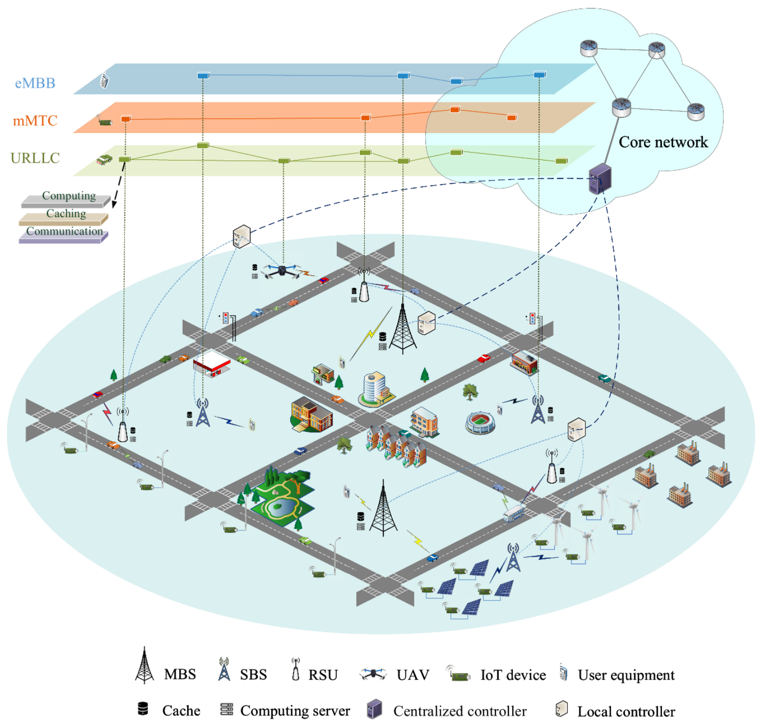

Figure 1: An illustration of network-slicing based NGWN architecture,with three example network slices. Reference: [4]

3 Resource allocation

Resource allocation is a crucial aspect of network slicing[1], as it determines how the available resources are allocated among the different slices. In network slicing, resources such as bandwidth, processing power, memory, and storage are divided and allocated to different slices according to their requirements and priorities. By efficiently allocating resources to each slice, network slicing can enable network operators to provide differentiated services to different customers and applications while maximizing the utilization of network resources.

Choosing the right allocation scheme is not as easy as it sounds. The provider has to balance the cost of reallocation with the QoS of service that needs to be provided to each slice.

Here are some examples of allocation schemes that can be used:

- Static : Allocate a fix resource from the start.
- Timed Re-slicing : Reallocate periodically using a timer.
- New Connections : Reallocate on new Connections.
- All Events : Reallocate on every event.

In the next part we will be presenting a simulator we implemented simulating *static* and *timed re-slicing.*

3.1 Simulation

The simulation is based on the theoretical model defined in paper [3]. Here we only aim to **maximise data rate**. **Note** that we could craft a model to ensure other QoS as latency, data security etc...

3.1.1 Model

We consider the downlink transmission of a 5G base station (BS) with a virtualized RAN and network slicing. We assume that there are S instantiated slices at the BS and denote their set by $\mathscr{T}$, $|\mathscr{T}| = S$. $C_s \geq 0$ denote the capacity of slice $s \in S$, so that

$$\sum_{s \in S} C_s \leq C, \tag{1}$$

The column vector of data rates is denoted by $\mathbf{R}[S \times 1] = (R_s)s \in S$. It is assumed that the infrastructure provider (InP) leases parts of its infrastructure in the form of slices to tenants. A Service Level Agreement (SLA) between the InP and the tenant includes the following slice characteristics:

- a minimum average user data rate $0 < R_s^{\min} \leq R_s$
- a maximum average user data rate $R_s \leq R_s^{\max} \leq C$
- a guaranteed capacity share γ_s or contracted number of users N_{cont}.

We assume that performance isolation of slice s is provided as long as

$$N_s \leq N_s^{cont}, \text{ or equivalently, } \frac{N_s R_s^{\min}}{C} \leq \gamma_s,\ 0 \leq \gamma_s \leq 1. \tag{2}$$

By performance isolation we mean that traffic fluctuation in one slice does not negatively affect performance in other slices.

3.1.2 Slicing scheme

The calculation of slices capacities is performed according to the paper [2]. We consider the vector N defined in the previous section. We consider the space $\Omega = \mathbb{N}^S$:

$$\Omega = \Omega_{\max} \cup \Omega_{\text{opt}} \cup \Omega_{\text{cong}}. \tag{3}$$

- In Ω_{max} : The usage is very light so we output the maximum throughput.

  For $N \in \Omega_{\text{max}} = \{N \in \Omega : NR^{\text{max}} \leq C\}$, we set:

$$R(N) = R_{\text{max}}, \quad s \in \mathscr{T} \implies C(N) = N_s R_s^{\text{max}}, \quad s \in \mathscr{T},\ N \in \Omega_{\text{max}}. \tag{4}$$

- In Ω_{opt} : The usage is moderate so we solve an optimisation problem to determine the capacity rates.

  For $N \in \Omega_{opt} = \{N \in \Omega : NR^{\text{min}} \leq C < NR^{\text{max}}\}$, we determine the data rates as the solution to the convex programming problem :

$$\max_{R \in \mathbb{R}^S} \quad U(R) = \sum_{s \in S} W_s(N_s) N_s \ln(R_s) \tag{5}$$

$$\text{s.t.} \quad NR = C, \tag{6}$$

$$\text{over} \quad R \in \mathbb{R}_+^S : R_s^{\text{min}} \leq R_s \leq R_s^{\text{max}}, \tag{7}$$

  where $W_s(N_s)$ is given by:

$$W_s(N_s) = \begin{cases} 1, & \text{if } N_s \leq N_s^{cont} \\ \frac{N_s^{cont}}{N_s}, & \text{if } N_s > N_s^{cont} \end{cases} \tag{8}$$

  The objective function (5) is differentiable and strictly concave by assumption and the feasible region is compact and convex, there exists hence a unique maximum for the data rate vector R_s, which can be found by Lagrangian methods.

- In Ω_{cong} : The usage is heavy so we assign the lowest possible value without impacting isolation.

  For $N \in \Omega_{cong} = \{N \in \Omega : NR^{\text{min}} > C\}$, we denote $N_s^{\text{min}}(N) = \min\{N_s, N_s^{cont}\}_{s \in \mathscr{T}}$

$$C_s(N) = \begin{cases} \frac{N_s^{\text{min}} R_s^{\text{min}}}{N^{\text{min}} R^{\text{min}}} C, & \text{if } N^{\text{min}} R^{\text{min}} \geq C, \\ N_s^{\text{min}} R_s^{\text{min}} + \frac{(N_s - N_s^{\text{min}}) R_s^{\text{min}}}{(N - N^{\text{min}}) R^{\text{min}}} (C - N^{\text{min}} R^{\text{min}}), & \text{if } N^{\text{min}} R^{\text{min}} < C \end{cases} \tag{9}$$

3.1.3 Implementation

We implement this in python using the multiprocessing and multi-threading library. The code can be found on this repository: `https://github.com/Joe-Khawand/Network_Slicing`. The implementation is divided into 4 separate processes:

- Process 0 : Manages the re-slicing allocations.
- Process 1 : Launches the first network slice.
- Process 2 : Launches the second network slice.
- Process 3 : Monitors the values of N_s and C in the slices to generate the graphs seen below.

The simulation of users is achieved through the use of threading in processes 1 and 2. To achieve this, each user is represented by a thread, enabling the system to efficiently handle multiple users concurrently. Additionally, the code has been developed using an object-oriented approach, providing users with the ability to customize the simulation parameters to meet their specific needs. This approach not only enhances the flexibility and versatility of the system but also ensures that the simulation accurately reflects real-world scenarios.

3.1.4 Simulation Parameters

We ran our simulation with those parameters:

- Number of slices : 2
- Simulation time : 60 seconds
- Central Capacity : 20
- Re-slicing time: $\frac{simulation\_time}{10}$

We consider two slices with different characteristics.

For slice 1 we have :

- $Rmin = 0.1$
- $Rmax = 7$
- $\gamma = 0.3$
- $\lambda_{adist} = 2$
- $\lambda_{sdist} = 35$

For slice 2 we have :

- $Rmin = 1$
- $Rmax = 1.5$
- $\gamma = 0.7$
- $lambda_{adist} = 1$
- $lambda_{sdist} = 1$

In summary, we can categorize Slice 1 as a user slice, designed for users who download large packages and have a flexible slice data rate. On the other hand, Slice 2 is tailored for an important machine that generates multiple small requests of small size.

3.1.5 Results

We obtained the following results showed in figure 2 and 3 :

Legend of the graphs :

- From top to bottom in both figures we plot the user data rates and the number of active users of slice 1 and 2. These are plotted as a function of the simulation time that can be found in the x-axis.
- The blue and red lines represent slice 1 and 2 respectively.
- The green and orange lines represent R_s^{min} and R_s^{max} respectively.
- The vertical yellow lines represent the re-slicing triggers.

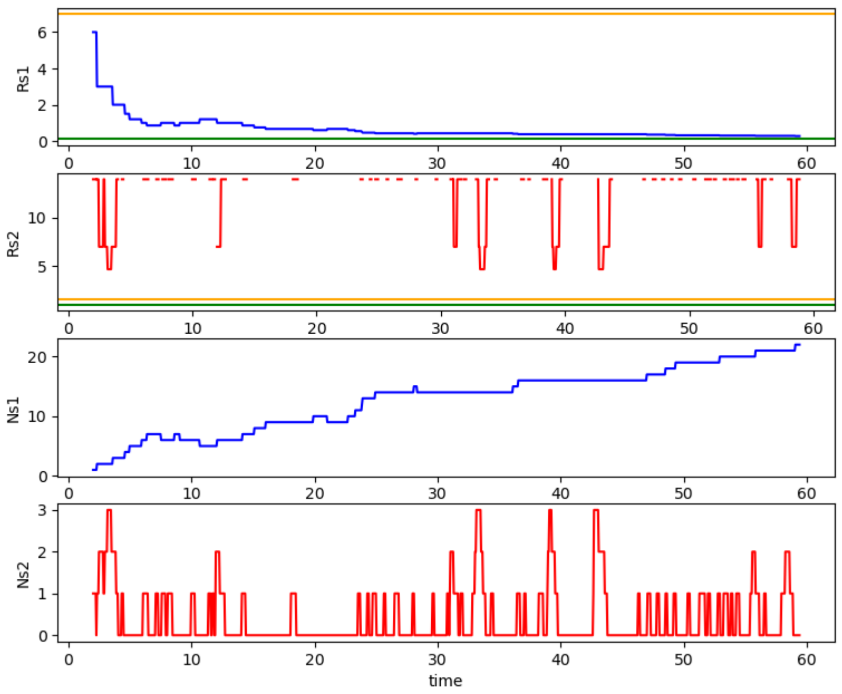

Figure 2: Simulation using static allocation.

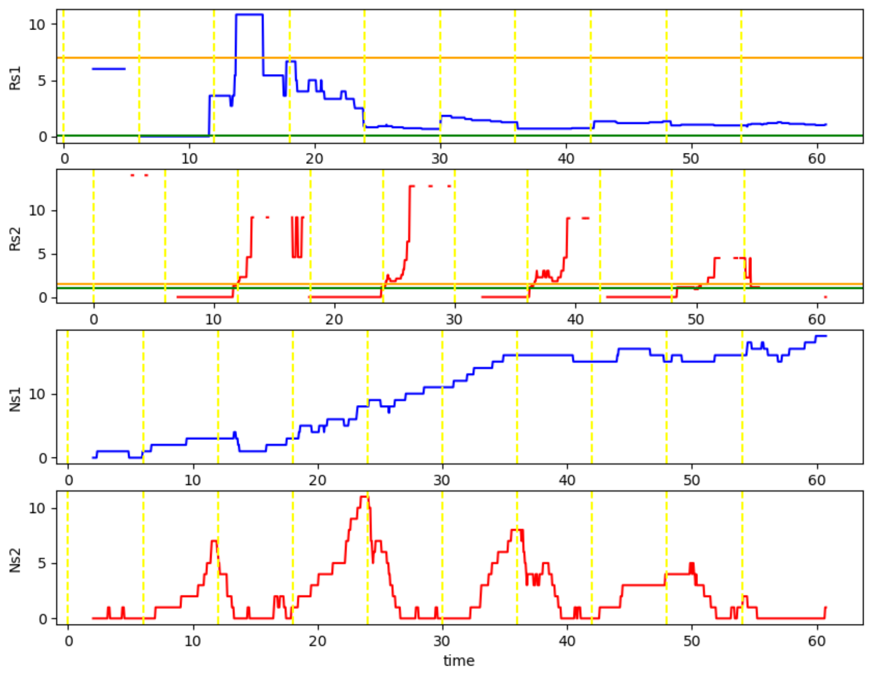

Figure 3: Simulation using static timed re-slicing allocation.

These can be interpreted as follow:

- With these characteristic we see that in a static simulation the performance of slice 1 is heavily affected. The slice is only able to send 6 files, whereas in a timed re-slicing simulation we send 18 files.
- The number of files sent are almost similar for slice 2 but we notice that slice 2 has worse latency than in slice 1 (normal because the objective function does not include latency).
- We notice that in the timed re-slicing simulation the SLA agreement is violated multiple times. The user rate goes below the orange Rmin line multiple times signaling degradation.
- On the other hand we notice that for a few brief moments after reallocation we verify the SLA agreements. We thus conclude that a dynamic reallocation scheme could solve that problem as showcased in the paper [2]. (but it is not necessarily the best solution because dynamic reallocation comes at a cost in energy usage, processing time, and processing power)

4 Conclusion

In conclusion, we have explored the promising concept of network slicing, and our simulation of two allocating schemes has provided a solid foundation for testing additional types of slices and slicing schemes with different objective functions. However, we cannot overlook the current drawbacks of this technology. The use of SDN technology introduces a single point of failure to the system, which can be a significant vulnerability if the SDN controller breaks down. Additionally, virtualizing functions and software can limit the performance that can be achieved on hardware and is often inferior to dedicated hardware. We must also consider the energy implications of maintaining such an architecture, as reslicing could be costly and may not align with the challenges of climate change. Despite these drawbacks, with further development, network slicing could prove to be a valuable tool for optimizing network performance and meeting the demands of various applications.

References

[1] Xiaoli Chu Keping Long Abdol-Hamid Aghvami Haijun Zhang, Na Liu and Victor C. M. Leung. Network slicing based 5g and future mobile networks: Mobility, resource management, and challenges. *IEEE Communications Magazine*, 2017.

[2] L. M. Correia N. Yarkina, Y. Gaidamaka and K. Samouylov. An analytical model for 5g network resource sharing with flexible sla oriented slice isolation. *Mathematics*, vol.8, 2020.

[3] Konstantin E. Samouylov Nikita A. Polyakov, Natalia V. Yarkina. A simulator for analyzing a network slicing policy with sla-based performance isolation of slices. 2021.

[4] JIE GAO 1 (Member IEEE)-WEN WU 1 (Student Member IEEE) KANGJIA LYU1 (Student Member IEEE) MUSHU LI 1 (Student Member IEEE) WEIHUA ZHUANG 1 (Fellow IEEE) XU LI2 XUEMIN SHEN1 (Fellow, IEEE) and JAYA RAO2. Ai-assisted network-slicing based next-generation wireless networks. *IEEE Open Journal of Vehicular Technology*, 2019.

[5] Shunliang Zhang. An overview of network slicing for 5g. *IEEE Wireless Communications*, 2019.