

COST-EFFECTIVE AND DDOS-AWARE NETWORK FUNCTION SHARING
MODELS FOR NEXT-GENERATION NETWORKS

by

Ömer Zekvan Yılmaz

B.S., Computer Science and Engineering, Sabancı University, 2006

M.S., Computer Science and Engineering, Sabancı University, 2009

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering
Boğaziçi University

2025

ACKNOWLEDGEMENTS

As my PhD journey comes to a close, I reflect on the many people who have supported me along the way. I would like to express my gratitude to my colleagues at TÜBİTAK, who encouraged me to embark on this journey, and to TÜBİTAK Scientist Support Programs Presidency (BİDEB), whose support through the 2211-National Graduate Scholarship Program made this research possible.

Embarking on my PhD journey at Boğaziçi University marked a significant turning point in my life, and it was there that I met my beloved wife, Sena. Her presence made this entire experience more bearable and enjoyable. I am truly grateful for her unwavering support.

I have also felt the constant encouragement of our families and friends throughout this long and challenging process. My sincere thanks go to my colleagues at ULAK Communications Inc., Dr. Selçuk Keskin and Dr. Emin Tarakçı, for their insightful reviews and ongoing support. I am equally thankful to all my colleagues and team members whose contributions made it possible to balance my PhD work with a busy schedule.

I would especially like to thank my advisor, Prof. Fatih Alagöz, who has provided invaluable guidance throughout the most difficult phases of this journey. I am also grateful to Prof. Tuna Tuğcu and Assoc. Prof. Şerif Bahtiyar for their valuable feedback throughout my thesis work. Finally, my heartfelt thanks go to all members of my thesis defense committee—Prof. Ertan Onur and Assist. Prof. Atay Özövde—for their constructive feedback and support.

ABSTRACT

COST-EFFECTIVE AND DDOS-AWARE NETWORK FUNCTION SHARING MODELS FOR NEXT-GENERATION NETWORKS

As next-generation communication networks evolve to meet increasingly diverse service requirements, optimizing resource efficiency and ensuring sustainability have become paramount. Network Function (NF) sharing has emerged as a promising solution for achieving both environmentally friendly and cost-effective resource allocation by maximizing resource utilization. However, maintaining Quality of Service (QoS) continuity and addressing security vulnerabilities together with NF sharing remain significant challenges. This thesis introduces a framework to optimize NF sharing for next-generation networks, with a focus on balancing cost efficiency with QoS and security. We propose a set of cost-efficient heuristics, CNFSH, along with its delay-aware version, D-CNFSH, to effectively manage the allocation and sharing of NFs. By prioritizing NF sharing, our approach significantly reduces underutilized resources, lowers operational costs, and supports green network initiatives, all while ensuring QoS continuity through the use of a priority-aware load balancer (LB). A novel metric, the ratio of underutilized to allocated resources, demonstrates the effectiveness of this method, with results showing underutilization reduced from around 30% in NoShare schemes to less than 0.4% with D-CNFSH. Additionally, the thesis addresses the security concerns associated with NF sharing by introducing SliceScore, an NF sharing-aware DDoS filtering mechanism designed to mitigate resource exhaustion threats. Simulations indicate that SliceScore outperforms traditional filtering methods by leveraging slice-specific insights to enhance defense strategies. These contributions lay the foundation for designing sustainable, secure, and cost-efficient communication networks in next-generation systems and pave the way for further research in the area.

ÖZET

YENİ NESİL HABERLEŞME AĞLARI İÇİN MALİYET ETKİN VE HİZMET REDDİ SALDIRILARINA DUYARLI AĞ FONKSİYONU PAYLAŞIM MODELLERİ

Yeni nesil ağlar, çeşitli hizmet gereksinimlerini karşılamak üzere dönüşürken, kaynak verimliliği ve sürdürülebilirlik kritik hale gelmiştir. Ağ Fonksiyonu (NF) paylaşımı, kaynak kullanımını maksimize ederek maliyet etkin tahsis sağlayan umut verici bir yaklaşımdır. Ancak, NF paylaşımıyla hizmet kalitesi devamlılığını sağlamak ve güvenlik açıklarını gidermek kritik bir sorundur. Bu tez, maliyet etkinliği ile servis kalitesi ve güvenlik arasında denge kurarak yeni nesil ağlar için NF paylaşımını optimize eden bir çerçeve sunmaktadır. CNFSH ve gecikmeye duyarlı D-CNFSH sezgisel yaklaşımlarını önererek, NF paylaşımını etkinleştirip kullanılmayan kaynakları minimize eden, operasyonel maliyetleri düşüren ve çevre dostu ağ çözümlerini destekleyen yöntemler sunuyoruz. Önerdiğimiz "kullanılmayan/tahsis edilen kaynak oranı" metriği, yaklaşımımızın verimliliğini göstermektedir. Sonuçlar, paylaşımsız yöntemlerde %30'a varan kullanılmayan kaynak oranının, D-CNFSH ile %0.4'ün altına indiğini ortaya koymaktadır. Tez ayrıca, NF paylaşımının güvenlik risklerini ele alarak DDoS saldırılarına duyarlı SliceScore filtreleme mekanizmasını sunmaktadır. Simülasyonlar, SliceScore'un dilime özgü bilgileri kullanarak geleneksel filtreleme yöntemlerinden daha iyi performans gösterdiğini kanıtlamaktadır. Bu çalışma, yeni nesil ağlarda sürdürülebilir ve güvenli haberleşme sistemleri tasarlamak için bir temel sunmakta ve daha verimli çözümler geliştirilmesine öncülük etmektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	iv
LIST OF FIGURES	viii
LIST OF TABLES	xi
LIST OF SYMBOLS	xii
LIST OF ACRONYMS/ABBREVIATIONS	xiv
1. INTRODUCTION	1
2. BACKGROUND	5
2.1. 5G Core Networks	5
2.2. Cloud-Native Network Function Orchestration	7
2.3. Network Slicing	8
2.4. Network Function Placement	9
2.5. Network Function Sharing	11
3. GREEN and DELAY-AWARE NF SHARING	12
3.1. Introduction	12
3.2. Problem Definition	15
3.2.1. Availability Constraint	15
3.2.2. NF Sharing Constraint	16
3.2.3. Bandwidth Sharing Constraint	17
3.2.4. Delay Constraint	18
3.2.5. Computing Capacity Constraint	18
3.2.6. Number of Compute Nodes Constraint	19
3.2.7. Slice Satisfaction Constraint	19
3.2.8. Objective function	20
3.3. The Proposed Heuristics and Algorithms	20
3.3.1. Sorting Computing Nodes by Ascending Remaining Capacities (RC)	21

3.3.2. Sorting Network Slice Requests by Ascending Requested Resources (RR)	21
3.3.3. Sorting Computing Nodes by Descending Remaining Capacities for Lower Delay (D)	23
3.3.4. Priority-Aware LB	26
3.3.5. Open-Source Implementation: SliceScheduler	29
3.4. Performance Evaluation	30
3.4.1. Performance Evaluation of the Proposed Heuristics	30
3.4.2. Performance Evaluation of the Priority-Aware LB	39
3.5. Conclusions	42
4. SLICESCORE: A SLICE ORIENTED DDoS FILTERING APPROACH . .	43
4.1. Introduction	43
4.2. The Attack Vectors	44
4.2.1. Cross-Slice Attacks	45
4.2.2. DDoS Attacks	46
4.2.3. Side-Channel Attacks	49
4.2.4. Exploiting Missing Security Zones	50
4.2.5. Elevation of Privilege Attacks	51
4.2.6. Exploiting Configuration Mistakes	52
4.2.7. Discussion on Previous Solutions	53
4.3. Previous Work in DDoS Filtering: ScoreForCore	54
4.4. Design and Operation of SliceScore	55
4.5. Performance Evaluation	62
4.6. Conclusions	73
5. CONCLUSION	75
5.1. Future Work	75
REFERENCES	77

LIST OF FIGURES

Figure 2.1.	Network Slicing.	6
Figure 3.1.	Traditional NF Placement with HA.	12
Figure 3.2.	Flow chart for priority-aware LB.	27
Figure 3.3.	Example allocation of underutilized resources and a pod failure scenario.	28
Figure 3.4.	The logarithmic scale of Average Execution Time (ms) vs. Number of Slice Requests.	32
Figure 3.5.	Number of SSRs without delay-aware heuristic vs. number of slice requests.	33
Figure 3.6.	Number of SSRs with delay-aware heuristic vs. number of slice requests.	34
Figure 3.7.	Percentage of change in the number of SSRs when the delay-aware heuristic is applied vs. number of slice requests.	35
Figure 3.8.	Resource Usage without Delay-Aware Heuristic vs. Number of Slice Requests.	36
Figure 3.9.	Resource Usage with Delay-Aware Heuristic vs. Number of Slice Requests.	37

Figure 3.10.	The percentage of underutilized to allocated resources vs. number of slice requests.	38
Figure 4.1.	SliceScore.	58
Figure 4.2.	Packet Processing in SliceScore.	60
Figure 4.3.	NS-3 simulation settings with specific victim scenario.	63
Figure 4.4.	Precision in 400 Mbps Attacks.	66
Figure 4.5.	Precision in 1.2 Gbps Attacks.	66
Figure 4.6.	Recall in 400 Mbps Attacks.	67
Figure 4.7.	Recall in 1.2 Gbps Attacks.	67
Figure 4.8.	True Negative Rate in 400 Mbps Attacks.	68
Figure 4.9.	True Negative Rate in 1.2 Gbps Attacks.	68
Figure 4.10.	Negative Predictive Value in 400 Mbps Attacks.	69
Figure 4.11.	Negative Predictive Value in 1.2 Gbps Attacks.	69
Figure 4.12.	Accuracy in 400 Mbps Attacks.	70
Figure 4.13.	Accuracy in 1.2 Gbps Attacks.	70
Figure 4.14.	F-Measure in 400 Mbps Attacks.	71

Figure 4.15. F-Measure in 1.2 Gbps Attacks.	71
Figure 4.16. F-Measure Complement in 400 Mbps Attacks.	72
Figure 4.17. F-Measure Complement in 1.2 Gbps Attacks.	72



LIST OF TABLES

Table 3.1. Delay measurements for Scaling and Fail-Over Scenarios with Priority-Aware LB. 41

Table 4.1. Attack Vectors Against NF Sharing Mentioned in the Literature . 48

Table 4.2. Comparison of DDoS Filtering Techniques. 56

Table 4.3. Key Performance Indicators (KPIs). 65

LIST OF SYMBOLS

A_f^{si}	Whether $f \in F$ is actively used by $s \in S$ as a host or guest for the index i of its requested NFs list.
$Av(x)$	Availability function for a given object x .
$B(L_{mn})$	Bandwidth capacity of the link L_{mn} .
$C(n)$	CPU capacity of node $n \in N$.
$D(L_{mn})$	Propagation delay of the link L_{mn} .
F	Set of NF deployments.
f_n	The pod of NF deployment $f \in F$ allocated to node $n \in N$.
L_{mn}	The link between two nodes; $m, n \in N$.
N	Set of compute nodes available for pod placement.
N_{max}	The number of compute nodes available.
$P(x)$	Failure probability of x .
R_s	Required redundancy for NFs of slice s to satisfy RAv_s .
RAv_s	Service availability required by the network slice $s \in S$.
RC_f	Requested number of CPUs by the NF f .
RD_s	The maximum delay allowed by the network slice $s \in S$.
RF_s	Set of NFs requested by the network slice s .
RT_s	Traffic flow requirement of the network slice $s \in S$.
S	Set of network slices.
$ScorePair^s$	The pair used in score calculations for $s \in S$.
SCP^s	Single Current Profile of $s \in S$.
SNP^s	Single Nominal Profile of $s \in S$.
$SPCP^s$	ScorePair Current Profile of $s \in S$.
$SPNP^s$	ScorePair Nominal Profile of $s \in S$.
$SuspiciousPair^s$	The attribute pair with the most probable signs for current attack to $s \in S$.
Th^s	The threshold for $s \in S$.
$TPCP^s$	The total number of packets in the current profile of $s \in S$.

$TPNP^s$	The total number of packets in the nominal profile of $s \in S$.
$OwnPair^s$	OwnPair of $s \in S$.
$OPNP^s$	OwnPair Nominal Profile of $s \in S$.
x_n^f	Whether the NF f is deployed on compute node $n \in N$.
y_n	Whether the compute node $n \in N$ is up and running.
z_s	Whether $s \in S$ is a satisfied slice request.



LIST OF ACRONYMS/ABBREVIATIONS

3GPP	3rd Generation Partnership Project
5GCN	5G Core Networks
AI	Artificial Intelligence
AY	Accuracy
CDF	Cumulative Distribution Function
CNF	Cloud-Native Network Function
CNFSH	Cloud-Native Network Function Sharing
CNFSH-RC	CNFSH with Requested Capacity heuristic
CNFSH-RCRR	CNFSH with both RC and RR heuristics
CNFSH-RR	CNFSH with Remaining Resources heuristic
CU	Central Unit
D-CNFSH	Delay-Aware CNF Sharing
D-CNFSH-RC	D-CNFSH with Requested Capacity heuristic
D-CNFSH-RCRR	D-CNFSH with both RC and RR heuristics
D-CNFSH-RR	D-CNFSH with Remaining Resources heuristic
D-NoShare	Delay-Aware heuristic without sharing
DDoS	Distributed Denial of Service
DNSH	DNS amplification attack on the highest capacity slice
DNSL	DNS amplification attack on the lowest capacity slice
DNSR	DNS amplification attack on random victims
DU	Distributed Unit
eMBB	enhanced Mobile Broadband
FM	F-Measure
FMC	F-Measure Complement
gNB	gNodeB, the base station for 5G
HA	High Availability
ITU-R	Radiocommunication Sector of International Telecommunication Union

LB	Load Balancer
mMTC	massive Machine-Type Communications
NF	Network Function
NFV	Network Function Virtualization
NGMN	Next Generation Mobile Networks Alliance
NoShare	NF placement without sharing as a benchmark
NPR	Negative Predictive Rate
PLMN	Public Land Mobile Network
PN	Precision
QoS	Quality of Service
RA	Random Attack on random victims
RAH	Random Attack on the highest capacity slice
RAL	Random Attack on the lowest capacity slice
RC	The heuristic sorting compute nodes in terms of remaining capacity
RR	The heuristic sorting slice requests in terms of requested resources
RL	Recall
RAN	Radio Access Network
SCA	Side Channel Attacks
SFC	ScoreForCore [1]
SDN	Software Defined Networking
SLA	Service Level Agreement
SSR	Satisfied Slice Request
TNR	True Negative Rate
UPF	User Plane Function
URLLC	Ultra-Reliable Low-Latency Communications
USR	User Service Request
VM	Virtual Machine
VNF	Virtual Network Function

1. INTRODUCTION

The rapid evolution of next-generation communication networks, driven by the increasing demand for diverse and high-quality services, has brought new challenges and opportunities for network optimization. As the need for higher performance and more flexible network architectures grows, the concepts of resource efficiency and sustainability have become central to the design of these networks. Traditional network designs are increasingly being reconsidered in favor of more dynamic, resource-sharing models that can enhance efficiency, reduce environmental impact, and lower operational costs.

Network slicing plays a pivotal role in enabling these dynamic models by dividing a single physical network infrastructure into multiple virtualized and independent slices, each tailored to meet the specific requirements of different services or applications. This approach allows service providers to deliver diverse services with distinct QoS guarantees while optimizing the underlying resources. However, the adoption of network slicing also introduces challenges, such as the need for efficient resource allocation, dynamic slice management, and maintaining inter-slice isolation to prevent performance degradation across slices. Security concerns, particularly those arising from virtualization and shared infrastructure, further complicate the realization of robust and reliable network slicing.

As networks become increasingly automated and intelligent, the role of Artificial Intelligence (AI) in managing and optimizing network operations is expanding. AI-driven automation is accelerating the deployment of new NFs, allowing networks to support a broader set of use cases and more complex operations. This surge in AI-assisted NFs introduces greater flexibility but also increases the difficulty of coordinating resources efficiently. The growing number of NFs, driven by machine learning-based traffic analysis, predictive maintenance, and autonomous decision-making, demands innovative strategies for effective NF placement and security.

One critical challenge within this context is the placement of network slices on a shared infrastructure. This problem involves determining the optimal allocation of resources to accommodate various network slices, each with its own QoS requirements, while minimizing resource usage and operational costs. Achieving an effective placement is complicated by the diversity of slice demands, which can range from ultra-low latency for mission-critical applications to high throughput for bandwidth-intensive services. Without efficient mechanisms to manage resource allocation, network operators risk underutilizing the available infrastructure, leading to inefficiencies and increased costs. This challenge highlights the need for advanced techniques like NF sharing to maximize resource utilization. By allowing multiple slices to share underutilized NFs, NF sharing not only addresses the inefficiencies of static resource allocation but also provides the flexibility to adapt to changing demands dynamically.

NF sharing is emerging as a promising strategy to further enhance resource utilization. By enabling multiple network slices to share common NFs, NF sharing provides significant cost savings while supporting the development of environmentally friendly, green networks. This approach leverages reserved but underutilized resources across slices, optimizing resource usage to achieve better cost efficiency and reduced latency. However, despite its operational benefits, NF sharing introduces unique security challenges, such as vulnerabilities to cross-slice interference and Distributed Denial of Service (DDoS) attacks, which demand innovative mitigation strategies.

This thesis addresses these challenges by proposing a comprehensive framework to optimize NF sharing, specifically targeting 5G and beyond communication networks, often referred to as next-generation communication networks, representing the evolution of modern telecommunications. We use both of the statements interchangeably in this thesis. Through the introduction of a novel delay-aware and cost-efficient heuristics, the research demonstrates how NF sharing can be effectively implemented to minimize resource underutilization, reduce costs, and support the realization of sustainable green networks. Furthermore, this work tackles the critical issue of security in NF sharing

by introducing SliceScore, a DDoS filtering mechanism that not only enhances network security but also ensures the efficient use of resources while preserving energy efficiency.

We describe a sustainable network as a communication infrastructure designed to efficiently utilize resources, minimize environmental impact, and ensure long-term operational viability while maintaining high performance, scalability, and security. Jo *et al.* [2], Saberi *et al.* [3], and Seuring *et al.* [4], who focus on sustainable smart cities, sustainable supply chain through blockchain, and supply chain management, respectively; all underline the environmental impact and the functionality when considering the sustainability of a system. Likewise, as the proposed mechanisms aim for a cost-effective, environmentally friendly, and secure network function sharing, the work in this thesis is an effort to improve the sustainability of next-generation communication networks.

This study is divided into two parts which are explained in Chapter 3 and Chapter 4. Chapter 3, explains our proposed NF sharing models that increase cost-efficiency, lowers resource usage, and provides delay sensitivity for network slice deployments. We propose CNFSH-RC, CNFSH-RR, and their combination to increase the number of satisfied slice requests and lower resource consumption. We refer to these heuristics as CNFSH altogether. Moreover we propose another heuristic, D-CNFSH, which incorporates delay constraints of network slices into the NF placement problem. The resulting model satisfies delay constraints of network slices but uses slightly more resources compared to CNFSH models, thus introducing a trade-off and a decision parameter in this aspect.

In Chapter 4, we propose SliceScore, an NF sharing aware slice-oriented DDoS filtering mechanism. First, we introduce new attack vectors that arise solely from NF sharing. We express the limitations of existing solutions in addressing these vulnerabilities and introduce SliceScore, which extends a previous DDoS filtering mechanism, ScoreForCore (SFC) [1], by evaluating network slice features to eliminate slice-targeted attacks.

Consequently, our contributions in this thesis are the following:

- (i) A combination of NF sharing models considering availability, bandwidth and delay constraints of network slices is devised.
- (ii) A set of heuristics, CNFSH, D-CNFSH and their combinations are proposed. As a result, an algorithm that converts the NP-Hard problem into polynomial time complexity, is devised and an open-source implementation is provided [5].
- (iii) A priority mechanism that maintains high availability (HA) for high-priority slices using a load balancer (LB) is proposed.
- (iv) To our knowledge, the proposed LB implements the first performance isolation mechanism for NF sharing.
- (v) We evaluate the performance of D-CNFSH using 10 core and 2 Radio Access Network (RAN) NFs. To our knowledge, D-CNFSH is the first NF sharing mechanism that targets 12 realistic 5G NFs, compared to previous studies such as [6,7].
- (vi) A new metric, the ratio of underutilized to allocated resources, is proposed.
- (vii) The attack vectors that arise solely by NF sharing are described.
- (viii) SliceScore, which incorporates the characteristics of the network slices into the score and the threshold calculations for more effective DDoS filtering compared to a previous high-performance solution, SFC [1], is proposed.

By presenting innovative solutions to the resource utilization and security challenges associated with NF sharing, this thesis lays the groundwork for the design of more efficient, secure, and sustainable communication networks. The findings provide valuable insights into the future of network optimization and contribute to the development of next-generation communication systems that can meet the evolving demands of modern society.

The remainder of the thesis is organized as follows: Chapter 2, gives a background on related topics. Chapter 3, introduces our NF sharing models. Chapter 4, explains SliceScore in detail. Finally, Chapter 5, concludes the thesis.

2. BACKGROUND

2.1. 5G Core Networks

The 5G Core Network (5GCN) serves as the brain of 5G networks, managing a wide range of critical functions, including billing, slicing, authentication, and authorization. Its architecture is inherently designed to operate in cloud environments, leveraging advanced paradigms such as network slicing, Software Defined Networking (SDN), Network Function Virtualization (NFV), and microservices architecture. These principles enable the 5GCN to provide flexible, scalable, and efficient network services tailored to diverse requirements.

A distinguishing innovation of 5G technology is its ability to cater to numerous vertical sectors while utilizing the same physical hardware resources. The International Telecommunication Union Radiocommunication Sector (ITU-R) [8] has identified three primary use cases that exemplify this capability: Enhanced Mobile Broadband (eMBB): Focused on providing high data rates for applications like video streaming and virtual reality. Ultra-Reliable Low-Latency Communications (URLLC): Critical for applications requiring near instantaneous communication and high reliability, such as autonomous vehicles and industrial automation. Massive Machine-Type Communications (mMTC): Designed for IoT scenarios involving a large number of connected devices, such as smart cities and agriculture. Each of these use cases presents unique requirements for throughput, latency, device density, and reliability. To address these varying demands, 5G employs network slicing, which creates multiple virtual networks on the same physical infrastructure. As illustrated in Fig. 2.1, network slicing enables service providers to allocate resources dynamically and cost-effectively, ensuring that different QoS levels are met for each slice.

Additionally, the adoption of microservices and containerization in the 5GCN allows for modular and independent deployment of NFs. This flexibility supports rapid

scalability and fault isolation, enhancing both performance and resilience. However, achieving optimal resource allocation across these slices remains a complex challenge, particularly when attempting to balance performance, cost, and energy efficiency.

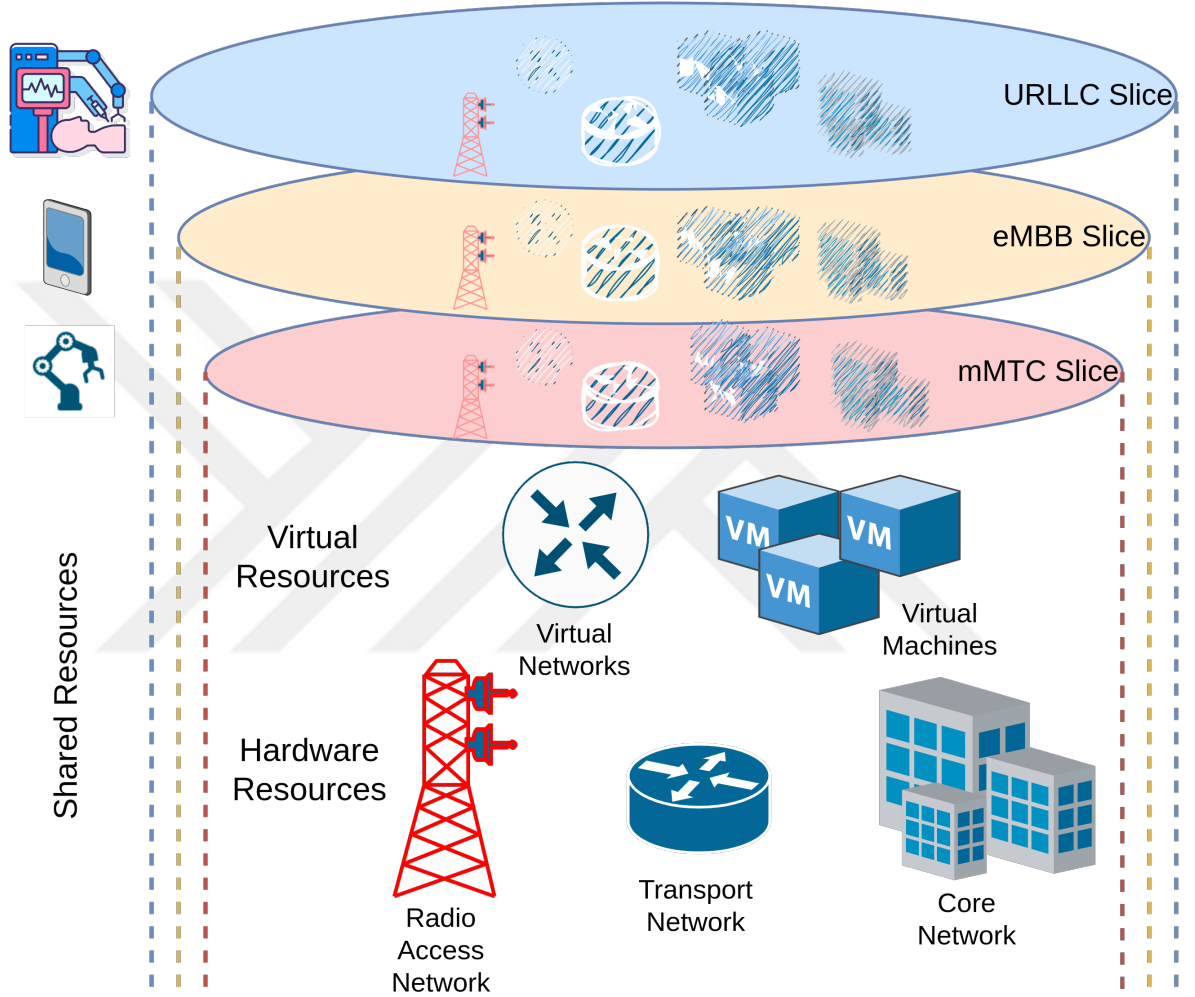


Figure 2.1. Network Slicing.

The evolution of 5G core networks is tightly coupled with advancements in AI and automation. AI is increasingly being integrated into network operations, enabling dynamic optimization, intelligent traffic management, and proactive security mechanisms [9, 10]. As AI-driven orchestration and automation mature, they will not only enhance existing NFs but also introduce entirely new NFs that are currently unforeseen. These emerging NFs will cater to specialized use cases, such as autonomous networks, AI-powered service assurance, and real-time adaptive security.

The expansion of NFs will naturally lead to an increase in the number of network slices. With 6G and beyond, network slicing is expected to evolve further, supporting highly customized slices for AI-driven applications, industrial automation, immersive extended reality, and URLLC. As a result, the number of slices deployed in future networks will scale significantly, leading to more complex and dynamic slicing requirements. This trend highlights the need for robust orchestration frameworks and enhanced security mechanisms to manage the growing interdependencies between NFs and slices effectively.

2.2. Cloud-Native Network Function Orchestration

The concept of 'cloud-native' refers to an architecture created especially for cloud settings. CNFs employ technologies such as microservices, containerization, and orchestration to realize such an architecture [11], allowing dynamic deployment and scaling. An orchestration platform called Kubernetes [12,13] ensures dependable and accessible CNF execution. The QoS criteria for CNFs include HA, low latency, and high bandwidth. While these specifications have to deal with issues that include jitter, network congestion, and all kinds of failures. Chapter 3 focuses on an appropriate load balancing and scaling architecture for CNFs to address these problems.

We give the following definitions in order to clarify some of the terms used in this thesis. A group of containers in Kubernetes is referred to as a pod, which includes a container that serves as an atomic microservice and other containers that handle secondary tasks like managing and maintaining the main container. A service object in Kubernetes serves as a gateway to a number of identical pods. The service object has four different types: ClusterIP, NodePort, LB, and External Name [14,15]. The pods and LBs are among the key components of our proposed mechanisms, CNFSH and D-CNFSH.

2.3. Network Slicing

The aim of 5G is to provide communication services for vertical sectors in a common infrastructure to reduce costs and operational workload [16]. This can be enabled through the adoption of concepts such as SDN, NFV, and Network Slicing [17, 18].

SDN enables the programmability of the user plane through central control. NFV is a paradigm that promotes the softwarization of network devices, enabling the abstraction and flexible management of resources. Although both provide network hardware abstraction, SDN deals with automating network management, and NFV focuses on softwarizing network functionalities.

In network slicing, the infrastructure is divided into logical segments, i.e., network slices (Fig. 2.1). Each network slice is designed to accommodate a group of services with a common set of QoS requirements and mostly dedicated to a single tenant. In this sense, apparently, a network slice should be isolated from other network slices in terms of security and performance [19].

Fig. 2.1 shows that the corresponding slices of 5G use cases, namely eMBB, URLLC, and mMTC, reside in the portions of the underlying resources: radio access network (RAN), transport network, core network, and even virtual machines (VMs) and virtual networks. The hardware and software resources available are shared among the network slices. Plenty of research and products are already available for hardware sharing. Research on cloud computing, NFV, and SDN focuses mainly on hardware abstraction [20, 21]. However, the sharing of software resources, such as NFs, has not yet been explored in depth.

In order to clarify network slicing and NF placement concepts, we give the example of smart factories. Smart factories rely on a multitude of applications with varying performance demands. Real-time machine vision inspection necessitates high through-

put and low latency for flawless video transmission, while remote robot control requires ultra-low latency for precise and near-instantaneous movement. Traditional networks cannot satisfy these requirements simultaneously.

Network slicing is a game-changing technology that allows virtual network slices to meet a variety of application requirements. To achieve the required performance criteria, each slice can be adjusted separately. Network slicing can be used in the smart factory scenario to generate various slices for different purposes.

CNFs are crucial in this transformation. Compared to hardware-based systems or even VMs, CNFs offer greater scalability and flexibility. The lifecycle of these CNFs, including deployment, scaling, and healing, is managed via CNF orchestration. The CNF orchestrator in the example of the smart factory would assign the right CNFs to each network slice. To effectively handle massive video streams, for example, high-throughput User Plane Functions (UPFs) would be implemented for the machine vision slices. To reduce signal delays, low-latency NF instances would be placed closer to robots in robot control slices. In order to maintain service continuity, the orchestrator would also automatically recover from failures and dynamically scale CNFs in response to traffic needs.

Smart factories can provide a dynamic and flexible network infrastructure that can handle a wide range of applications with different performance needs by using network slicing and CNF orchestration. This strategy opens the door to a more reliable and efficient future for smart manufacturing by ensuring service continuity through automated CNF management in addition to optimizing resource consumption.

2.4. Network Function Placement

NF placement has emerged as a critical area of research in the context of modern communication networks, particularly with the advent of 5G and beyond. The placement of NFs directly impacts the performance, scalability, and cost-efficiency of

network infrastructures. Effective placement strategies ensure that network slices and their associated NFs meet strict QoS requirements while optimizing the use of underlying resources. This dual objective of performance and efficiency necessitates advanced techniques that can adapt to the dynamic nature of network demands.

The complexity of NF placement arises from the need to balance multiple factors such as latency, throughput, resource utilization, and availability. As communication networks evolve, the integration of virtualization and containerization technologies further complicates this problem. Virtual Network Functions (VNFs) and Cloud-native Network Functions (CNFs) bring flexibility to NF deployment but also introduce challenges in maintaining performance isolation, managing redundant resources, and minimizing deployment costs.

A key challenge in NF placement is addressing underutilized resources within the network. These resources, often left idle due to static or suboptimal allocation strategies, represent a significant opportunity for improvement. By leveraging techniques such as resource sharing and dynamic allocation, network operators can enhance both resource utilization and overall network efficiency. This optimization is particularly important as the demand for diverse and resource-intensive services continues to grow.

In the NF placement domain, the problem is formulated as follows: Mostly a list of network slice requests is to be deployed on a limited number of computing nodes. Each network slice has a set of requirements, including a number of NFs that need to be in close interaction to carry out the services offered by the network slice. Therefore, the optimum placement of NFs on the underlying infrastructure is needed in order to satisfy the maximum number of network slices. Dietrich et al. [22] improve load balancing and network slice acceptance rates by using a linear programming approach for NF placement. Meanwhile, CloudScope was built on top of the Xen hypervisor by Chen et al. [23], who used an online Markov Chain model for online prediction of the performance of co-resident VMs. In order to lower deployment costs and improve QoS, Bushnaq et al. [24] propose an automatic network slicing optimization framework.

With the adoption of containerization, research on CNF placement has increased. Taking availability needs into account, Arora and Ksentini [25] propose a dynamic resource allocation approach. It appears that deploying redundant copies of the NFs is necessary to ensure availability as a result of placing the NF within QoS limitations. The concept of ‘underutilized resources’ arises from this correlation, suggesting the potential for further performance improvements.

2.5. Network Function Sharing

In order to reduce the costs of redundant resource provisioning in communication networks, NF sharing is proposed in recent studies [7, 26]. These studies investigate methodologies on how to share NFs among network slices to minimize costs while maintaining the functionality of individual network slices.

Virtual NF (VNF) sharing is used by Mohamad and Hassanein [7], resulting in 47% more satisfied requests than the model without sharing. The resource usages of the network slices are considered static in [7], and their dynamic nature is not taken into account. Certainly, it requires further resource management to make the transition to a dynamic environment, which we are aiming for in this thesis. Additionally, [7]’s solution is NP-hard, which will result in more delays in large-scale slice scheduling.

Truong-Huu et al. [26] have developed a more refined sharing-aware strategy that takes into account the traffic capacity and resource requirements of both shared and nonshared VNFs, based on the functions of each VNF. Meanwhile, Vittal and Franklin [27] use load balancing in kernel space to ensure the HA of network slices in 5G, but without NF sharing. In [27], performance was measured by the ratio of preventing the occlusion of User Service Requests (USR), achieving 60%, and the average response time of these USRs, which is reduced by 50%.

3. GREEN and DELAY-AWARE NF SHARING

3.1. Introduction

The growing need for applications with high throughput and low latency is driving service providers to improve NF placement methodologies in cloud settings with minimal costs. One of the main factors supporting cost-effective cloud computing is the sharing of NFs amongst network slices, which is already proposed by [7, 28–30]. The idea behind NF sharing is the largely underutilized resources due to QoS requirements, as shown in Fig. 3.1, while further exploitation of these resources is possible.

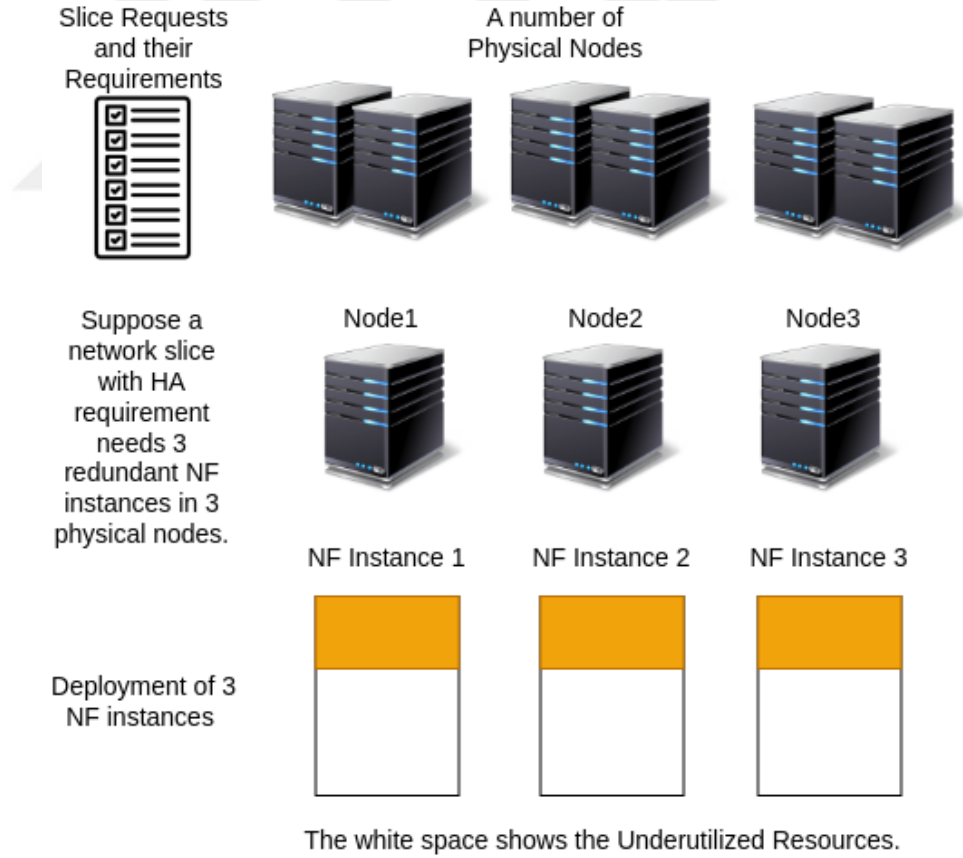


Figure 3.1. Traditional NF Placement with HA.

NF placement intends to deploy the NF in the optimum location such that QoS requirements are met in a cost-effective way. In order to deploy an NF with a low

delay requirement, shortest-path communication links should be established. Since the number of these shortest-path links is limited, maximizing resource utilization through sharing is an effective way to reach performance objectives.

The same applies to bandwidth requirements. Among the available resources to deploy an NF, there are locations that lack bandwidth capacity, while other locations in other computing nodes might provide higher bandwidth. Therefore, applying the appropriate NF placement policies is crucial for bandwidth requirements as well.

However, due to delay constraints, the resource consumption of NFs increases compared to delay-tolerant scenarios. The NFs in a chain are required to be placed in close proximity to each other so that communication overhead is reduced. In this study, we demonstrate how delay-aware services require more resources and how NF sharing reduces these requirements and helps to save energy.

Meanwhile, network slicing can be defined as the virtualization and sharing of hardware and software resources among users. This yields virtual networks, virtual storage, virtual CPUs, and virtualized NFs as well. In this thesis, we further virtualize NFs by sharing them among slices in order to increase resource utilization and achieve QoS goals with lower costs.

5G networks leverage NFV and SDN to gain flexibility and adopt a service-oriented approach. The use of these paradigms allows network slicing to satisfy a broad range of use cases, like high reliability for autonomous vehicles or ultra-low delay for remote surgeries. This requires highly scalable and reliable NFs. Detailed 5G architecture and specific NF requirements are discussed in Chapter 2 and in [6, 31–33].

Deploying 5G NFs in cloud-native environments is a cost-effective solution for mobile network operators. As mentioned by Arouk and Nikaein [11], containerization is one of the design principles of a cloud-native architecture. Frangoudis et al. [34] show that containers achieve lower response times due to having less interaction with the

operating system and network I/O operations compared to VMs. Moreover, in [34], it is shown that the start time of a container is around 1 second, while it takes around 11 seconds for a VM, an indication of the latency required for the scaling and failover scenarios in both cases. These pieces of evidence clearly show that CNFs surpass VNFs in satisfying 5G use cases.

One of the concerns about sharing of NFs is the security and privacy implications. Obviously, isolation among network slices and prevention against denial of service attacks, side channel attacks, etc. are necessary in the case of NF sharing [35,36]. Taking into account these security precautions, the sharing of 3G/4G/5G NFs has been devised in 3GPP releases [16] and other studies [6,37]. In order to address the security concerns related to DDoS attacks, our proposal SliceScore is presented in Chapter 4.

Moreover, performance isolation, which implies both the security precautions and the performance guarantees of individual slices, is another principle of network slicing. This aspect is mostly ignored or even sacrificed in previous studies. Zharabad et al. [6] use shared VNFs and shared CNFs for cost-effective 5G core networks. However, Zharabad et al. [6] propose to trade off isolation and sharing if necessary. In this thesis, we show that using a load balancing mechanism and orchestration capabilities in cloud-native environments, performance isolation is achieved with negligible costs.

In this chapter, we present the NF sharing models and their heuristics, CNFSH [38] and D-CNFSH [39], which use a priority-aware LB for placement and sharing of CNFs in next-generation networks. D-CNFSH considers the availability, performance isolation, and delay requirements of network slices at the same time. The advantage of shared NFs in satisfying these requirements together had not yet been adequately considered in the literature.

3.2. Problem Definition

In this section, we present the availability model, a bandwidth sharing model, a delay model, an NF sharing model, and the objective function. As in our case, the NF placement problem generally aims to maximize the number of satisfied slice requests (SSR) with a fixed amount of resources, taking into account the requirements of 5G and beyond.

Let S be the set of network slice requests and RF_s be the set of NFs requested by each $s \in S$. RAv_s is the availability requirement of s . RT_s is the traffic flow requirement of s . The set of NF deployments F includes all deployed and running NFs. Each $f \in F$ has a resource requirement RC_f . All deployments are made on a set of compute nodes N , and each $n \in N$ has $C(n)$ amount of resources.

3.2.1. Availability Constraint

The availability constraint [25] for NFs defines the acceptable downtime of an NF that can be tolerated before significantly impacting the functionality of the network slice. This constraint is crucial to ensure high reliability aspects of 5G use cases. Obviously, in order to cope with service level agreements (SLAs), network operators should prioritize resources, implement redundancy measures, and define recovery procedures.

Each computing unit, whether it is a physical machine, a VM, or a container, has a probability to fail for some time. $P(f)$ resembles the failure probability of an NF f . $P(n)$ resembles the failure probability of a node n . We assume that the availability of an NF instance (pod) relies on the success probabilities of the pod itself and of the physical node that hosts the pod. Therefore, the availability of a pod residing on node n is given by

$$Av(f_n) = (1 - P(n)) * (1 - P(f)^{x_n^f}), \forall f \in F, \forall n \in N, \quad (3.1)$$

where f_n is the NF pod residing on the compute node n . $Av(f_n)$ is the availability of f_n . x_n^f is a binary decision about whether to deploy f_n on compute node n or not.

If the availability of f_n is not sufficient for the availability constraint of f , then f should be instantiated with a sufficient number of pods to comply with this constraint. Hence, the product of failure probabilities of NF pods gives the failure probability of the NF as indicated by

$$Av(f) = 1 - \prod_{n \in N} (1 - Av(f_n)), \forall f \in F, \quad (3.2)$$

where $Av(f)$ is the availability of f .

All NFs required by a network slice should be up and running in order to consider the network slice available, which means that the product of the success probabilities of the required NFs gives the success probability of the network slice. Hence, the availability of a network slice s , $Av(s)$, is given by

$$Av(s) = \prod_i^{|RF_s|} \sum_{f \in F} A_f^{si} Av(f), \forall s \in S, \quad (3.3)$$

where A_f^{si} is a binary decision about whether the deployed NF f is actively used by s for the index i of RF_s .

Each network slice request comes with a set of requirements, one of which is the availability constraint. The availability constraint tells us that the actual availability of a network slice, $Av(s)$, should not be less than the required availability, RAv_s , as in

$$RAv_s \leq Av(s), \forall s \in S. \quad (3.4)$$

Otherwise, the network slice is not satisfied.

3.2.2. NF Sharing Constraint

Network slices are classified into high- and low-priority categories based on their HA requirements. Slices that cannot tolerate service failures longer than 1 second in containerized environments and 11 seconds in VM environments are considered high-priority, based on the measurement in Section 4.5. The relationship between HA requirements and high-priority classification is a one-to-one correspondence—any slice with such strict HA constraints is necessarily high-priority.

In CNFSH [38] and D-CNFSH [39], NF sharing leverages idle resources while ensuring that the HA of high-priority slices (hosts) remains protected through priority-aware load balancing. Ghosh and Banerjee [40] introduce a priority-aware LB mechanism that suspends low-priority traffic under congestion. Similarly, in our approach, if a pod failure occurs, the priority-aware LB dynamically reroutes host traffic to available pods, while low-priority guest slices are temporarily placed in a waiting queue. Guest slices are selected based on their ability to tolerate short delays and occasional packet loss until a new pod is deployed.

Frangoudis et al. [34] measured CNF deployment times to be approximately 1.129 seconds on a 16-core Intel Xeon processor with 32 GB of RAM, running Ubuntu 14.10. Considering this and our measurements in Sec. 4.5, we define the classification boundary between high- and low-priority slices. This classification ensures that NF sharing does not compromise the HA guarantees required by critical network functions.

Eq. 3.5 shows that only a limited number of network slices are able to use the NF f with the already deployed pods:

$$\sum_{s \in S} \sum_i^{|RF_s|} A_f^{si} \leq \sum_{n \in N} x_n^f, \forall f \in F, \quad (3.5)$$

which indicates that the number of network slices using the NF can not exceed the number of deployed replications for that NF. Deploying one NF instance for each NF requirement is actually similar to the classical NF placement methodologies. However, our proposal with a priority-aware load balancing approach favors the network slice with high priority to maintain its HA requirement. Thus, securing a sufficient number of NF pods in case of congestion or node failures.

3.2.3. Bandwidth Sharing Constraint

The bandwidth constraint of a directed link between two compute nodes should aggregate all directed flows between the pods residing at these compute nodes. Like most of the studies in the literature [7, 26, 41], we assume that the amount of flow

required by a network slice is identical between all the NFs involved. Hence,

$$\sum_{s \in S} \sum_{i=0}^{|RF_s|-1} \sum_{f \in F} \sum_{g \in F} RT_s x_m^f A_f^{si} x_n^g A_g^{s(i+1)} / R_s \leq B(L_{mn}), \forall m, n \in N, \quad (3.6)$$

determines whether the flow between consecutive NFs can be handled by the physical link, where R_s is the required redundancy for NFs of slice s to satisfy its availability constraint in Eq. 3.4. $B(L_{mn})$ is the link bandwidth capacity between nodes m and n . Since there is no direct bonding between RF_s and the deployed NFs, we need to traverse all deployed NFs twice to track the NF deployment tuples that are serving the network slice itself. The traffic flow required for an NF is divided by the number of redundant pods of that NF, so that the load is balanced and the underutilized bandwidth resources can be shared with other network slices.

3.2.4. Delay Constraint

The delay constraint of a network slice requires that the total delay of the flow that traverses all NFs of the network slice should not exceed a certain threshold. Therefore, whether the total delay is below the limit that the slice can tolerate is determined by

$$\sum_{i=0}^{|RF_s|-1} \sum_{f \in F} \sum_{g \in F} \max(D(L_{mn}) x_m^f A_f^{si} x_n^g A_g^{s(i+1)} | m, n \in N) \leq RD_s, \forall s \in S, \quad (3.7)$$

where $D(L_{mn})$ is the delay of the link between nodes m and n . RD_s is the delay budget of slice s . Here we calculate the maximum total delay that a slice can experience using the replicated pods of its NFs. We assume that $D(L_{mn})$ is constant and is not affected by the bandwidth usage of the same link. Therefore, sharing of NFs, load balancing, or any traffic congestion does not affect the delay constraint in our assumption. We also assume that the delay of incoming and outgoing traffic to and from the edge network will not affect the placement decisions in our scenarios.

3.2.5. Computing Capacity Constraint

A physical node n has a limited CPU capacity. Therefore,

$$\sum_{f \in F} x_n^f RC_f \leq C(n) y_n, \forall n \in N, \quad (3.8)$$

limits the total reservation of CPUs at node n ; where y_n is a binary decision for running the compute node n .

3.2.6. Number of Compute Nodes Constraint

In our formulation, we assume a limited number of compute nodes available for deployment. While the Computing Capacity Constraint (Eq. 3.8) ensures that the total CPU reservation at each node does not exceed its capacity, we must also explicitly limit the total number of active compute nodes. Let N_{max} denote the maximum number of compute nodes that can be utilized. Then, we introduce

$$\sum_{n \in N} y_n \leq N_{max}, \quad (3.9)$$

which ensures that the number of active compute nodes does not exceed the predefined limit, preventing excessive resource allocation beyond the available infrastructure.

3.2.7. Slice Satisfaction Constraint

In this study, we need to count the number of satisfied network slices that have been deployed with all of their NFs. A network slice is satisfied only if its required NFs are deployed and up and running. Hence,

$$z_s = \prod_{i=0}^{|RF_s|} \sum_{f \in F} A_f^{si}, \quad (3.10)$$

determines the state of the network slice s , which indicates that the slice is satisfied if z_s is 1; otherwise, it is 0. Eq. 3.10 checks all deployed NFs to see if the deployed NF f is allowed to be used by s for the index i of RF_s . The product of the outcomes for all the indices in Eq. 3.10 enforces an active and deployed NF for each index i of RF_s to consider a satisfied network slice.

3.2.8. Objective function

The objective function

$$\min \frac{1}{1 + \sum_{s \in S} z_s}, \quad (3.11)$$

is in favor of maximum number of SSRs, where the addition of 1 to the denominator is to avoid the division by zero problem.

In Section 4.5, we try to achieve the maximum number of SSR for a fixed number of physical nodes for each experiment. This indicates that each SSR has the same value for the objective function. As we see in Sec. 3.3, we propose a special heuristic (RR) in order to benefit from the value equality of the SSR.

3.3. The Proposed Heuristics and Algorithms

The above-mentioned problem is an example of the Multiple Multidimensional Knapsack Problem (MMdKP) [42] and can be solved with Integer Linear Programming with NP-Hard complexity [43], but it is not always feasible with a number of slice requests above 200, as experimented by Vimont et al. [44]. Panigrahy et al. [45], Ang et al. [46], and many others propose heuristics using various sorting techniques to tackle bin-packing and knapsack problems. In [38], we have proposed the CNFSH algorithm, where the $O(|S| \log |S|)$ time sorting function and the $O(|S|^2)$ time search for a suitable host NF make it feasible with polynomial-time complexity. In order to further investigate the behavior of CNFSH [38], we have adopted two heuristics, RR and RC. Additionally, in [39], we have proposed a heuristic for delay-aware NF placement and sharing, D-CNFSH.

In the following, we explain our proposals, the three heuristics, the priority-aware LB that contribute to the performance of NF placement and sharing, and our open-source implementation, SliceScheduler.

3.3.1. Sorting Computing Nodes by Ascending Remaining Capacities (RC)

One of the criteria in knapsack problems is to leave as little waste space as possible in the computing node after placing the NF [42]. The waste space is the residual resource of the computing node, such that it is too small to place a new NF. In this respect, the heuristic RC sorts the computing nodes in terms of remaining capacities in increasing order for each network slice request iteration, as shown in *Algorithm 1*.

3.3.2. Sorting Network Slice Requests by Ascending Requested Resources (RR)

The objective function in Eq. 3.11 assumes that all slice requests have the same value. Therefore, it is not worth placing a network slice with higher resource requirements instead of another slice with low resource requirements. It will consume more energy for the same profit. In this case, sorting slice requests in terms of the requested resources in increasing order is a viable solution for this scenario, as shown in *Algorithm 1*.

This scenario is applicable to the NF placement problem in 5G networks and beyond, as there are cases where any satisfied network slice yields the same profit for the network operator. In this case, RR will be a suitable heuristic to solve the NF placement problem.

Algorithm 1 Non-Delay-Aware Slice Scheduler.

```

if  $Scheme == RR$  then
     $SortByRequestedResources(S)$ 
end if
while  $s$  in  $S$  do
    while  $TRUE$  do
        if  $Scheme == RC$  then
             $SortByRemainingCapacities(N)$ 
        end if
        while  $r$  in  $RF_s$  do
             $Found \leftarrow false$ 
            if  $Scheme \neq NoShare$  and  $Priority(s) == Low$  then
                while  $f$  in  $F$  do
                    if  $Type(r) == Type(f)$  and  $UnderutilizedCapacity(f) \geq RC_r$  then
                         $Found \leftarrow true$ 
                         $break$ 
                    end if
                end while
            end if
            if  $Found == False$  then
                 $F.append(r)$ 
                 $Deploy(r, Requirements(r), s)$ 
            end if
        end while
    end while
end while

```

Algorithm 2 Deploy(NetworkFunction f , Slice s).

 $p \leftarrow \text{CalculateRequiredReplications}(f)$
while n in N **do**
if $\text{RemainingCapacity}(n) \geq \text{RequiredCapacity}(f)$ **and** $\text{CalculateDelaySoFar}(s) \leq RD_s$ **then**
 $\text{Place}(f)$
 $\text{UpdateRemainingCapacity}(n);$
 $p \leftarrow p - 1;$
if $p \leq 0$ **then**
 break
end if
end if
end while
 $\text{ReturnDeploymentResults}$

3.3.3. Sorting Computing Nodes by Descending Remaining Capacities for Lower Delay (D)

In addition to CNFSH [38], we have proposed a heuristic that helps delay-aware network slices to be deployed in 5G and beyond edge settings, which we call D-CNFSH. Most of the previous studies, like [7], use NP-Hard algorithms for delay awareness. D-CNFSH is composed of a delay-aware NF sharing model and a heuristic solution. The remaining capacities of the compute nodes are sorted in descending order for each network slice to be deployed, as shown in *Algorithm 3*. This is to support the deployment of the necessary NFs as close as possible so that the communication links between NFs result in a lower delay. In fact, its combinations with previous heuristics, D-CNFSH-RC and D-CNFSH-RCRR (D-RC) schemes, lose their advantage of sorting the nodes in the order of ascending remaining capacities. This is because RC and delay-aware sorting heuristics offer a trade-off between the number of SSR and delay, which can be used as a design parameter for a wide range of use cases, as explained below.

Algorithm 3 Delay-Aware Slice Scheduler.

```

if  $Scheme == RR$  then
     $SortByRequestedResources(S)$ 
end if
 $RepeatForLowDelay \leftarrow false$ 
while  $s$  in  $S$  do
    while  $TRUE$  do
        if  $Scheme \neq RC$  or  $RepeatForLowDelay$  then
             $SortByDescendingRemainingCapacities(N)$ 
        else
             $SortByRemainingCapacities(N)$ 
        end if
         $RepeatForLowDelay \leftarrow false$ 
        while  $r$  in  $RF_s$  do
             $Found \leftarrow false$ 
            if  $Scheme \neq NoShare$  and  $Priority(s) == Low$  then
                while  $f$  in  $F$  do
                    if  $Type(r) == Type(f)$  and  $UnderutilizedCapacity(f) \geq RC_r$  and
                         $CalculateDelaySoFar(s) < RD_s$  then
                         $Found \leftarrow true$ 
                        break
                    end if
                end while
            end if
            if  $Found == False$  then
                 $F.append(r)$ 
                 $Deploy(r, Requirements(r), s)$ 
                if  $DeploymentFailedDueToDelayConstraint$  then
                     $F.pop()$ 
                     $RepeatForLowDelay \leftarrow true$ 
                end if
            end if
        end while
    end while
end while

```

In *Algorithm 3*, we start with sorting the slice requests in terms of the requested resources in ascending order if it is an RR scheme. Then, we start processing the slice requests one by one. For each slice request, there will be a loop until we find that there is no space available. For an RC scheme, we sort the compute nodes in terms of the remaining capacities in ascending order. However, for schemes that are both delay-aware and RC, we follow a strategy that first looks for available resources as an RC scheme without delay awareness. If no resources are found, possibly due to a delay constraint, we sort the compute nodes by remaining capacities in descending order. This sorting is used to keep the NFs of the delay-constrained network slice in close proximity to achieve a low delay. A scheme that is delay-aware and not RC will directly sort the compute nodes in a delay-aware manner. As a result, only the schemes that are delay-aware and RC simultaneously loop twice; others loop only once.

Later, we loop for each NF f required by the network slice. Except for NoShare schemes, if the requested network slice has a low priority, we check all previously deployed NFs of the same type with f to see if we can find some underutilized resources that are enough for f . Then, if the established NF chain so far does not violate the delay constraint of the requested network slice (Eq. 3.7), the already deployed NF is shared and f is marked as satisfied. If no underutilized resource is found for f , then a new NF should be deployed.

The NF deployment algorithm shown in *Algorithm 2* takes the NF f and its requirements, such as availability, CPU, memory, and delay, and the requested network slice s , as parameters, and looks for an available compute node to host the f . The algorithm first calculates the number of required replications of f that are enough to satisfy the availability requirement. Then each compute node in the list is checked to see if it can provide the required resources. Later, assuming that f is placed at that node, the delay of the previously deployed NFs of the network slice s is calculated and compared with its delay constraint. In case these checks are successful, an NF pod is deployed in this compute node. The loop is executed until all necessary replications of f are successfully deployed.

3.3.4. Priority-Aware LB

In order to protect the availability requirements of high-priority slices that can be affected by node or pod failures and traffic congestion, as well as to maintain the performance isolation of network slices, we employ a priority-aware LB by extending [40]. The complete logic and flow chart of the LB is shown in Fig. 3.2. First, a state table that holds the current bandwidth usage is prepared for each deployed pod. Then, the LB starts to wait for USRs from clients. Each USR is checked to see whether it belongs to a high- or low-priority slice. If it belongs to a low-priority slice, then an additional check is performed to determine whether the requested NF is in an alarm state or not. If the alarm is active or there is no available pod, the request is sent to the waiting queue. In this way, the LB detects any congestion, suspends low-priority slices, and raises an alarm. This takes 0.105 milliseconds (ms) in the worst case, according to the measurements of Ghosh and Banerjee [40]. Otherwise, the low-priority slice is routed towards an available pod. On the other hand, if no available pod is found for the USR belonging to a high-priority slice, an alarm is raised, and the request is sent for reprocessing.

In a separate thread, the LB sends health checks to deployed pods. If a pod does not reply, the pod is marked as not available in the state table, and an alarm is raised. Alarms are also processed in a separate thread, such that SLAs are checked to see whether further resources can be allocated to the corresponding network slice. If this is the case, scaling can be triggered, and the alarm is resolved in case of successful scaling. If the scaling is not included in SLAs of the network slice, the corresponding USR can be dropped due to bandwidth overload. If the alarm is resolved, the USRs in the waiting queue are sent for reprocessing; otherwise, they drop out after a timeout period.

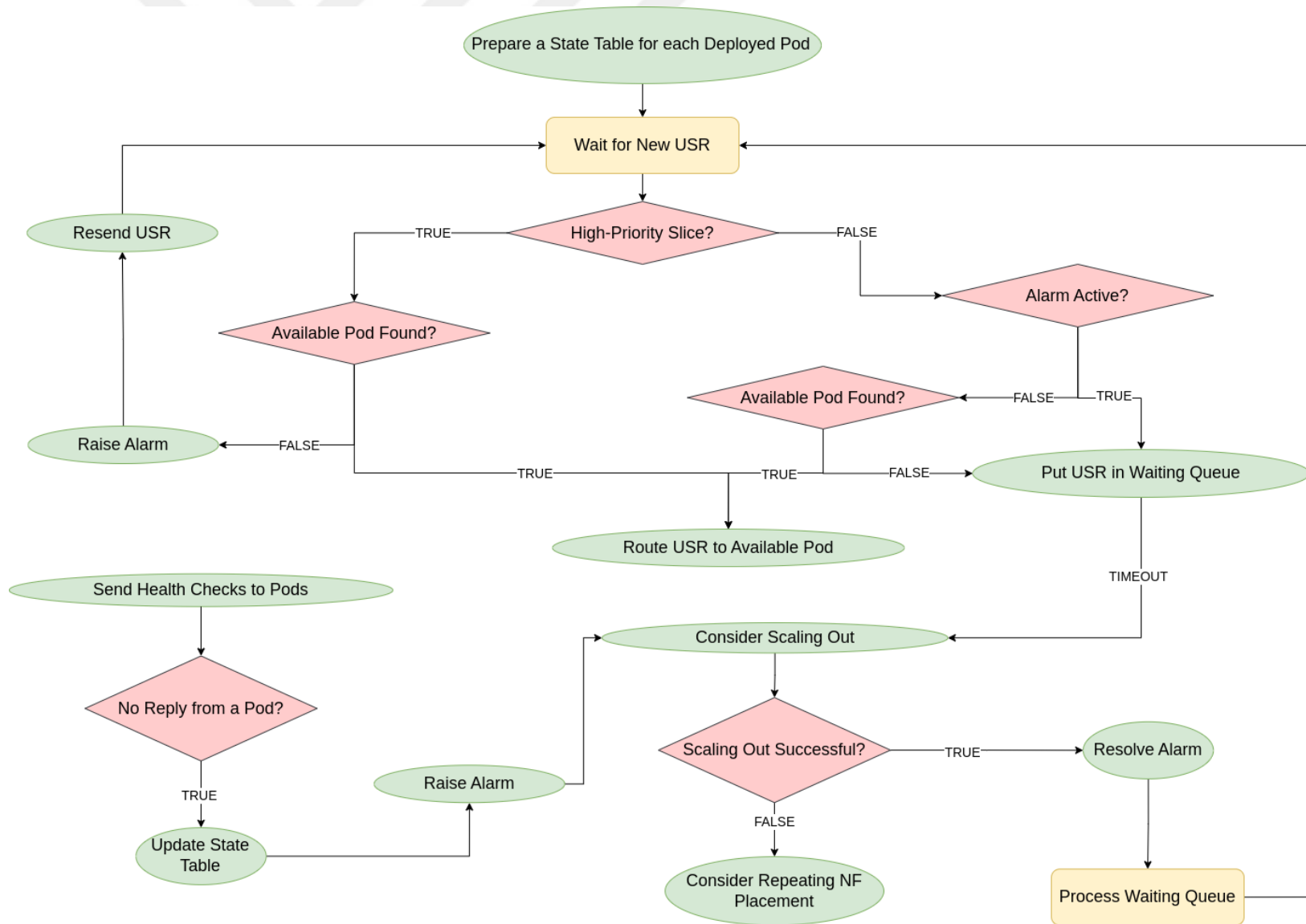


Figure 3.2. Flow chart for priority-aware LB.

It is worth mentioning that we deploy redundant pods in order to satisfy the availability constraints of the network slices, and we make use of the underutilized resources that result from this redundancy. In Fig. 3.3, we demonstrate some cases that could be encountered in the lifetime of these pods. In the first column of Fig. 3.3, we see that one network slice is using a couple of pods. The white space resembles the underutilized resources that are already allocated to these pods. The second column shows two new network slices that are making use of these resources.

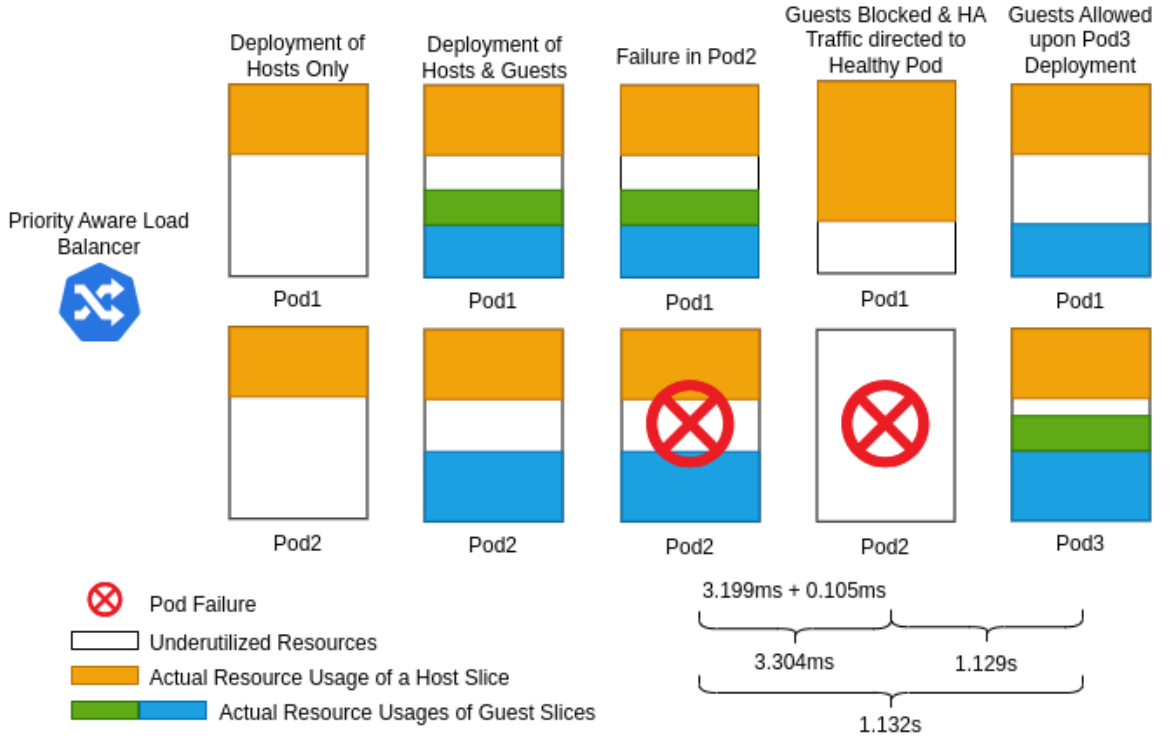


Figure 3.3. Example allocation of underutilized resources and a pod failure scenario.

Our measurements show that the round trip delay between LB and a Kubernetes pod is 1.199 ms in the worst case and 0.639 ms in the best case. If the LB sends health checks to deployed pods every 2 ms (a configurable parameter) in separate threads, a pod failure can be detected in 3.199 at worst case. Then an alarm is raised, and none of the low-priority USRs are allowed to reach the deployed NF pods, as shown in the third column in Fig. 3.3. Later, a new pod (Pod3) is triggered and instantiated in around 1.129 seconds [34], and corresponding USRs are directed to Pod3 in around 0.105 ms for each USR in the worst case [40], as shown in the fifth column (Fig. 3.3).

This also shows that the availability and performance isolation of these low-priority slices are not violated for more than around 1.132 seconds with D-CNFSH.

Noting that SLAs and the measurements of Frangoudis et al. [34] are used to determine the priorities of the network slices. Furthermore, if any congestion occurs due to a violation of the SLA limits, then predefined policies should be implemented, which can be detected using the LB state tables.

3.3.5. Open-Source Implementation: SliceScheduler

In order to disseminate the work of this thesis, in addition to our published work, we have provided the open-source implementation of SliceScheduler [5], which experiments twelve different models explained in Section 4.5.

The SliceScheduler is designed to allocate network slice requests efficiently across a limited number of nodes while ensuring optimal resource utilization. It iterates through multiple scheduling experiments, dynamically generating slice requests and applying different strategies across various control groups. Unlike traditional schedulers, it evaluates slice requests based on priority, the necessary constraints, incorporating NF-sharing mechanisms to reduce redundant deployments.

This scheduling approach is particularly relevant to Kubernetes, which, while widely used for cloud-native orchestration, lacks network-awareness and NF-sharing capabilities. The default Kubernetes scheduler primarily considers CPU and memory constraints without optimizing for telecom-specific needs such as delay sensitivity, NF sharing, and HA. The SliceScheduler enhances Kubernetes by introducing intelligent scheduling policies that maximize the number of allocated slices while reducing unnecessary resource consumption. By supporting NF sharing, it significantly improves efficiency, allowing slices to reuse existing network functions instead of deploying new ones.

Another key advantage is its ability to support HA and delay-aware scheduling, both critical for 5G and beyond computing. The scheduler ensures that latency-sensitive slices are placed in optimal locations to minimize delay while also considering redundancy requirements for service continuity. Unlike Kubernetes' default approach, which does not factor in network latency, the SliceScheduler strategically places workloads to enhance performance in real-time applications. Additionally, its monitoring mechanisms provide deeper insights into resource utilization trends, preventing both over-provisioning and underutilization.

By bridging the gap between Kubernetes' general-purpose scheduling and the specialized demands of network slicing, the SliceScheduler enables more scalable and cost-efficient NF deployment. It ensures that network slices are allocated optimally, balancing efficiency with performance requirements. With features like function-sharing, HA considerations, and latency-aware placement, it offers a more refined scheduling model for cloud-native NFV environments, making it an essential enhancement to Kubernetes-based telecom infrastructures.

3.4. Performance Evaluation

In this section, we first evaluate the performance of proposed heuristics and then the priority-aware LB.

3.4.1. Performance Evaluation of the Proposed Heuristics

For the performance evaluation of our proposed heuristics, we have extended our open-source slice scheduler on GitHub [5]. For comparison, we use six non-delay-aware schemes: two NoShare models, the dynamic version of the sharing model of Mohamad and Hassanein [7] as MSHS, and three CNFSH schemes: CNFSH-RC, CNFSH-RR, and CNFSH-RCRR; and six delay-aware schemes: D-NoShare, D-CNFSH, D-CNFSH-RC, D-NoShare-RR, D-CNFSH-RR, and D-CNFSH-RCRR. RC, RR, and D-CNFSH heuristics are explained in Sec. 3.3.

We use MSHH [7] for comparison because it aims to exploit all the underutilized resources at a given time t , yet in a static manner. In order to make the comparison possible in a dynamic setting, we adopt our priority mechanism into MSHH.

NoShare models do not use any NF sharing logic; NoShare-RR only sorts the slice requests in terms of ascending requested resource size. The remaining algorithms first sort the slice requests by priority. No more sorting is performed in MSHH. CNFSH-RC and CNFSH-RR apply the heuristics explained in Sec. 3.3. CNFSH-RCRR applies both of them. Delay-aware schemes, i.e., variations of D-CNFSH, follow *Algorithm 3*. The algorithms using RR are evaluated separately for fairness.

We use four metrics to compare all these schemes: the number of SSRs, the average CPU allocation per network slice, the computation time, and our proposed metric, the ratio of underutilized to all allocated resources. With the assumption that the memory requirements are in parallel with the computing requirements, we just use the CPU requirements in our calculations. The experiments were run on Ubuntu 20.04.6 running on an Intel Core i7-8550U CPU @1.80GHz and 16 GB of memory. An NF catalog, containing 10 sharable NF types as in [29] and the addition of 2 RAN functions, CU and DU, which are described by 3GPP [47], was used. We have also improved our open-source implementation [5] with faster execution times compared to [38], as seen in Fig. 3.4.

In the experiments, the slice requests consist of 3-10 different NF types selected randomly from the NF catalog. User Plane Function (UPF), which has a heavy workload as devised by the study in [48], requires 8 CPUs. Other NFs require 2 CPUs. High-priority slices, within the scope of URLLC, require 99.999% availability, as in NGMN [49]. Low-priority slices, within the scope of mMTC, require less than 98% availability. All NFs and physical nodes have 0.999 availability, as experimented by Arora and Ksentini [25].

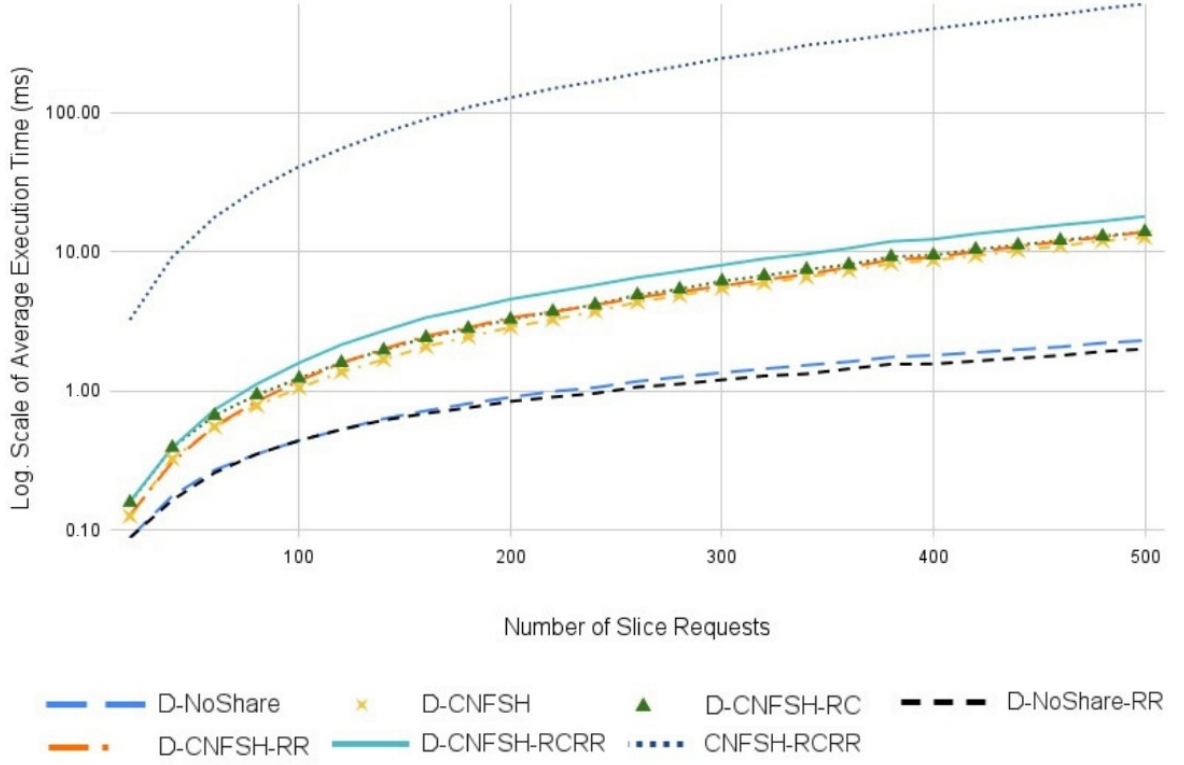


Figure 3.4. The logarithmic scale of Average Execution Time (ms) vs. Number of Slice Requests.

Various numbers of randomly generated slice requests are experimented with between 20 and 500 with a step size of 20. Each set of slice requests is tested for all twelve models. Each experiment is repeated 600 times, and the average values are used. Six physical nodes with decreasing capacities from 100 to 80 CPUs were simulated.

For delay-sensitive experiments, all network slices were assigned a random delay budget between 10 and 20 ms. The link delay between NFs in the same computing node is assumed to be 0, and between NFs in different computing nodes it is assumed to be 10 ms.

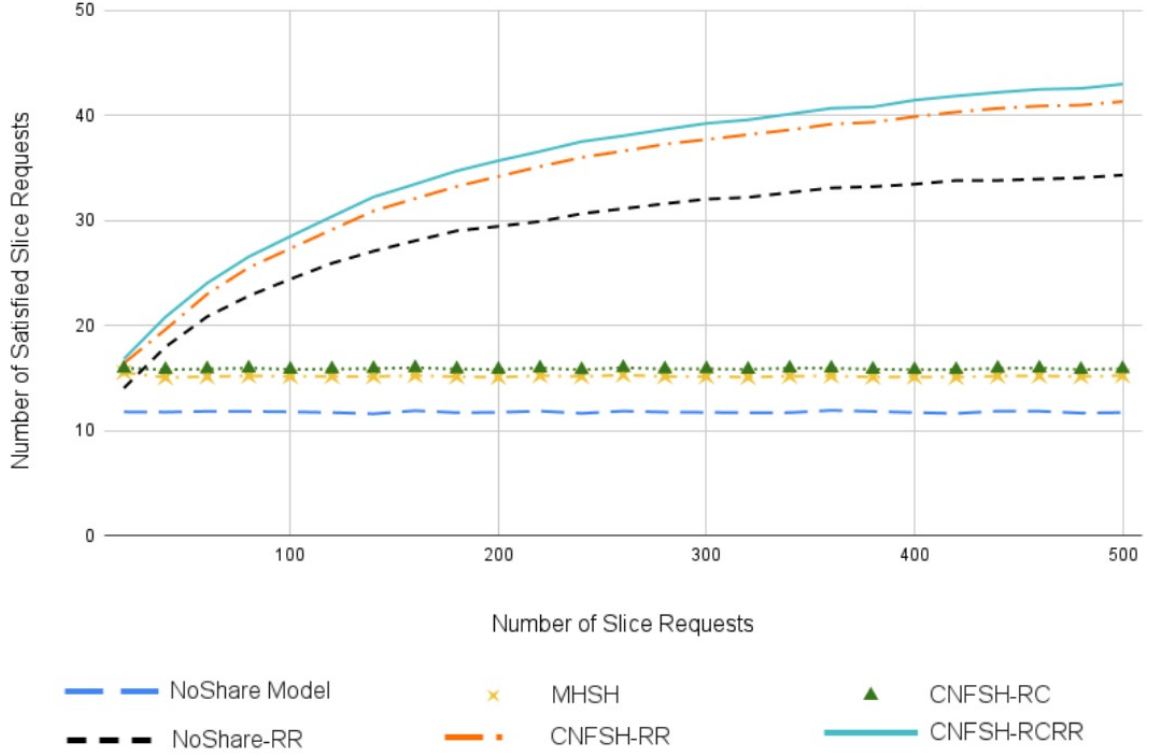


Figure 3.5. Number of SSRs without delay-aware heuristic vs. number of slice requests.

In Fig. 3.5, 3.6 the y-axes show the absolute number of SSRs. The x-axes in all figures show the number of slice requests. MSH and CNFSH-RC reach around 30% and 35% higher SSRs than the NoShare Model, respectively. At the same time, CNFSH-RC reaches 5% higher SSRs than the MSH model. CNFSH-RR and CNFSH-RCRR achieve 24% and 29% higher SSRs compared to NoShare-RR (Fig. 3.5).

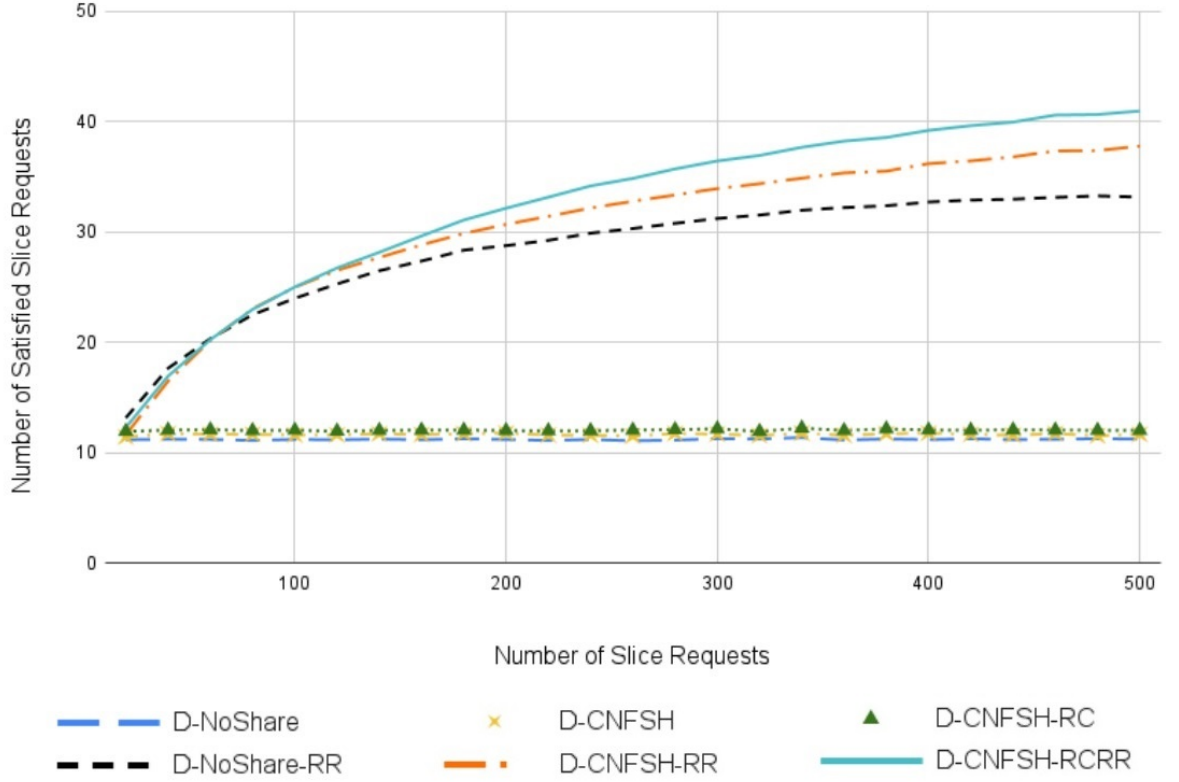


Figure 3.6. Number of SSRs with delay-aware heuristic vs. number of slice requests.

In Fig. 3.6, D-CNFSH-RC achieves around 2% higher SSRs than D-CNFSH when the sorting heuristic for delay awareness is applied. The advantage of D-CNFSH-RC is due to the proposed heuristic, which tries to find a balance between ascending and descending sorting of the remaining capacities of nodes in order to maximize SSRs with delay awareness.

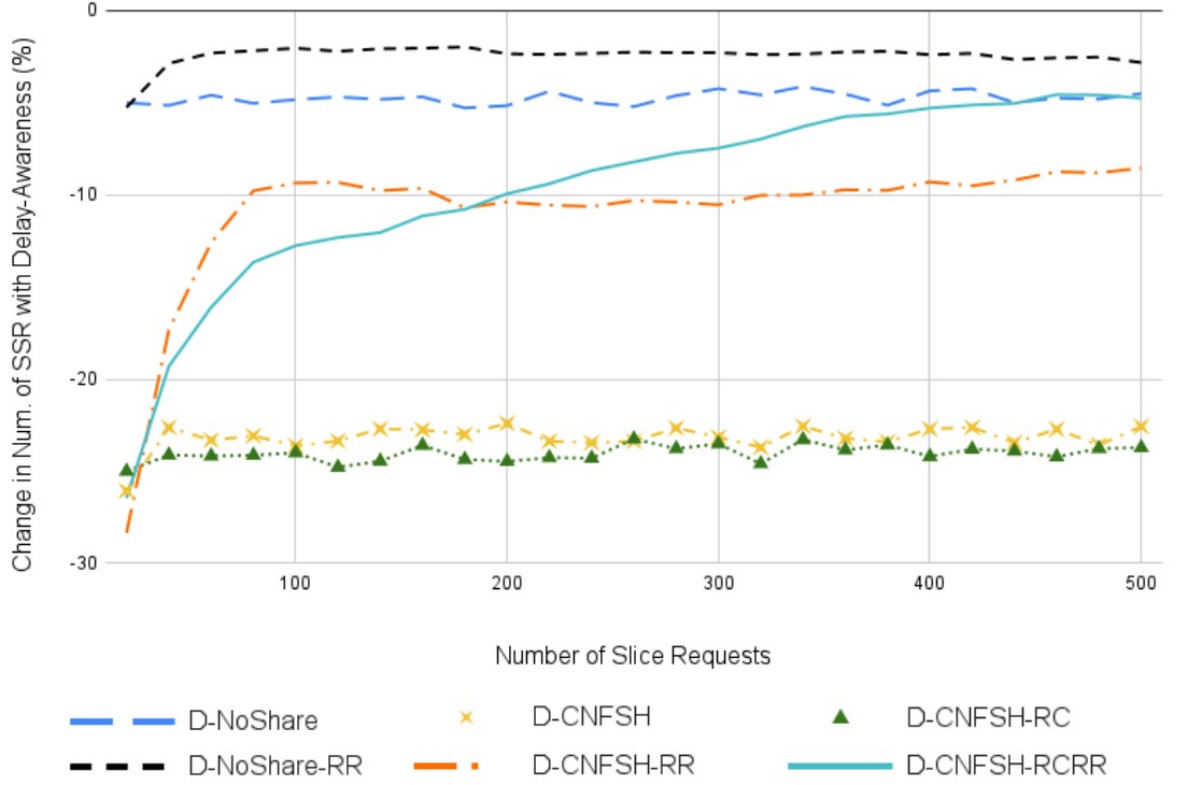


Figure 3.7. Percentage of change in the number of SSRs when the delay-aware heuristic is applied vs. number of slice requests.

In Fig. 3.7, the y-axis shows the percentage of change in the number of SSRs when the delay-aware heuristic is applied. D-CNFSH is considered the delay-aware version of MSHS. It is notable that RR schemes are less affected by delay constraints compared to CNFSH schemes without the RR heuristic, due to the fact that RR schemes prefer network slices with a low number of NFs, and these network slices are less likely to be rejected due to delay constraints.

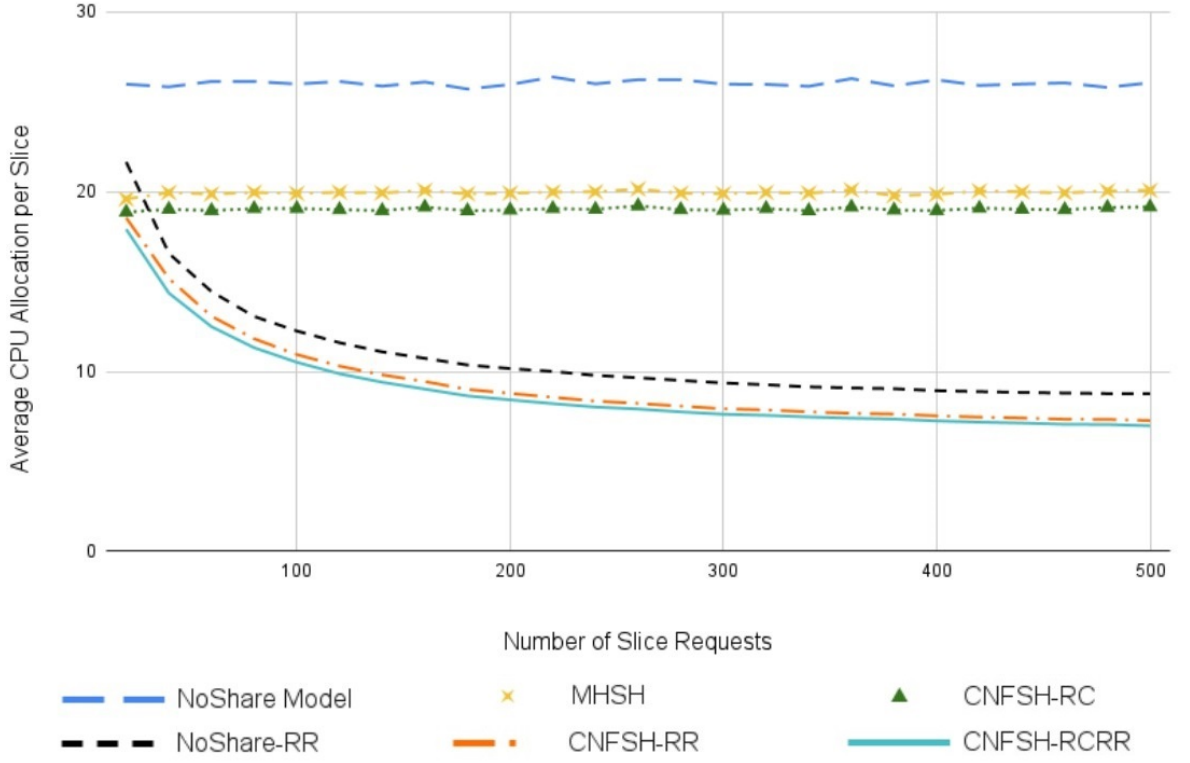


Figure 3.8. Resource Usage without Delay-Aware Heuristic vs. Number of Slice Requests.

In Fig. 3.8, 3.9, the y-axes show the average CPU allocated to each satisfied slice. CNFSSH-RC has 28% and 5% less resource consumption compared to the NoShare Model and MSHH, respectively. While CNFSSH-RCRR uses 20% less resources than NoShare-RR (Fig. 3.8). Resource usage increases when the delay-aware heuristic is applied, as seen in Fig. 3.8, 3.9. For instance, the CPU allocation to network slices increases by around 9% and 5% with D-CNFSSH-RR and D-CNFSSH-RCRR, respectively, compared to their delay-tolerant versions with 500 slice requests (Fig. 3.8, 3.9).

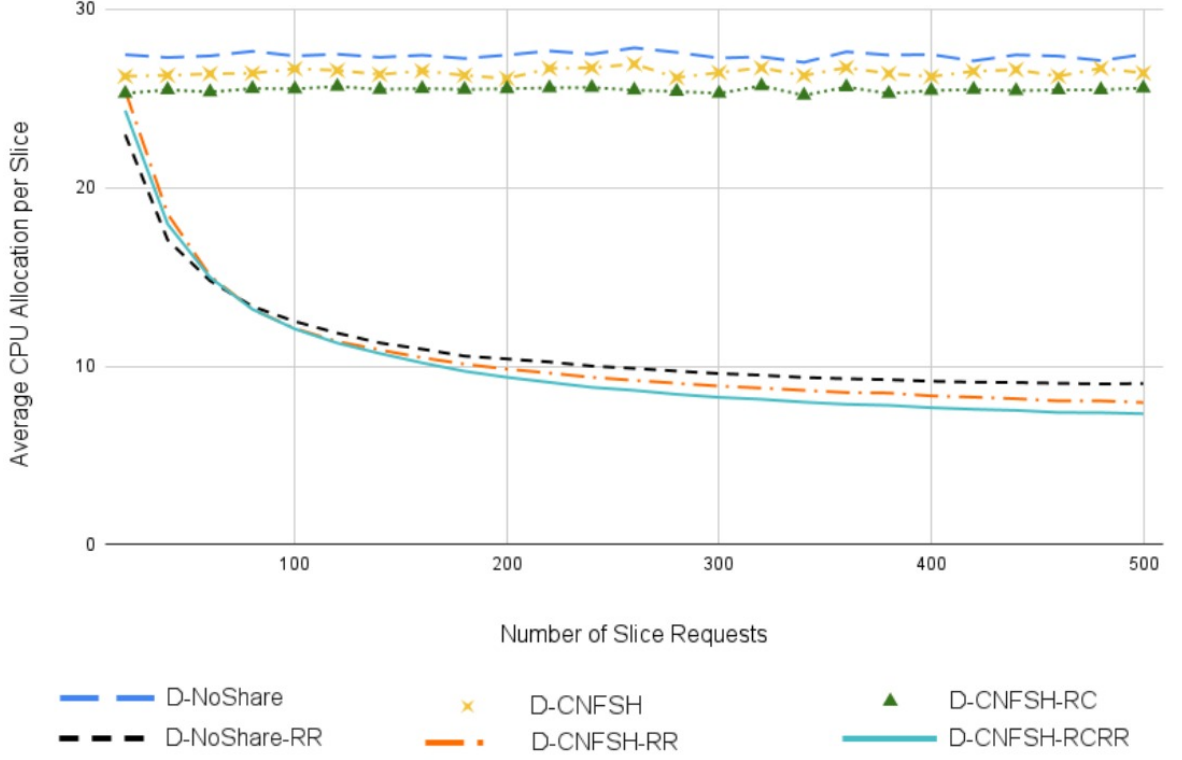


Figure 3.9. Resource Usage with Delay-Aware Heuristic vs. Number of Slice Requests.

In Fig. 3.10, our novel metric, the ratio of underutilized to allocated resources on the y-axis, is around 30% in the D-NoShare and D-NoShare-RC models and drops below 1.3% in all sharing models. Moreover, this ratio is around 0.4% with D-CNFSH, around 0.5% with D-CNFSH-RC, and around 1.2% with both D-CNFSH-RR and D-CNFSH-RCRR. The high ratio with RR schemes is due to the fact that in each slice request, the NF with the heaviest workload, UPF, can take place only once. UPF causes the highest amount of underutilized resources. As a result, because of the higher number of SSRs with RR, the ratio of underutilized to allocated resources becomes less without RR. Our proposed metric, the ratio of underutilized to allocated resources, investigates how much energy is wasted per energy used. In that sense, it is worth using this metric in environmentally sensitive and green applications.

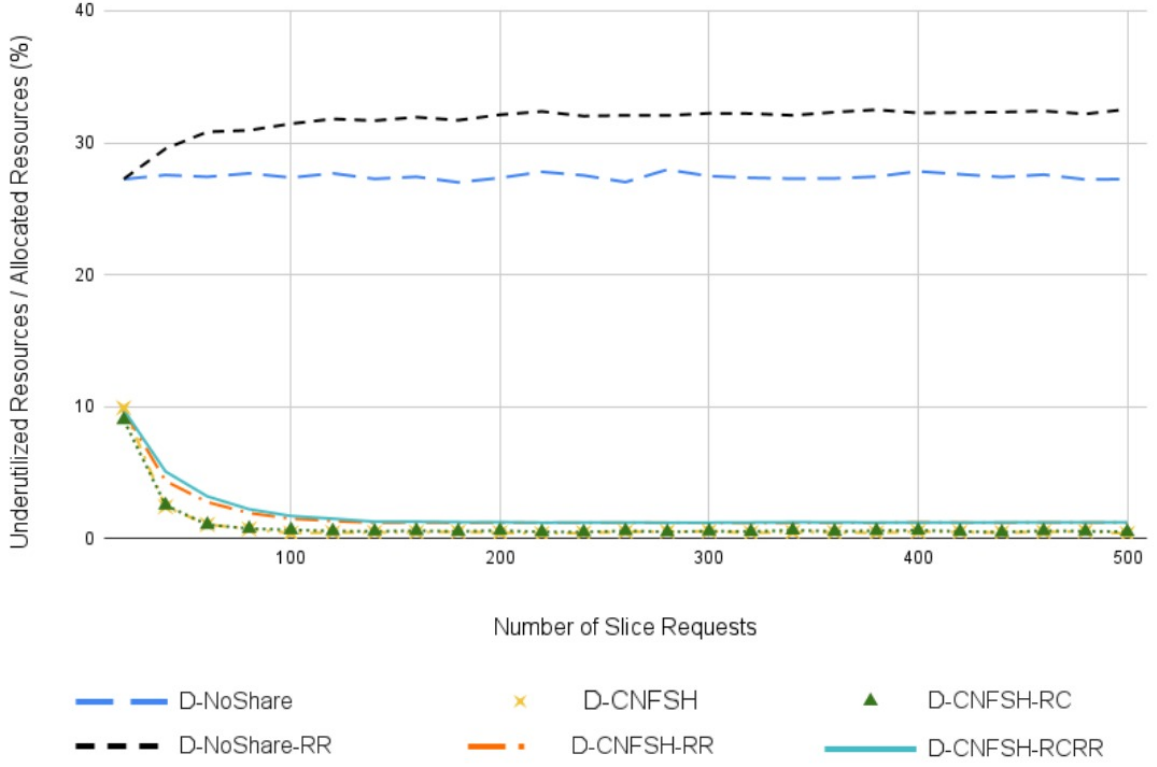


Figure 3.10. The percentage of underutilized to allocated resources vs. number of slice requests.

The experiments with delay-aware schemes show that deploying delay-aware slices comes with the cost of extra computing resources necessary to achieve the same number of satisfied requests with delay-tolerant ones. As seen in Fig. 3.5, 3.6, 3.7, the performance loss of D-CNFSH-RR and D-CNFSH-RCRR is around 9% and 5% compared to CNFSH-RR and CNFSH-RCRR, respectively, for 500 slice requests. However, in D-CNFSH and D-CNFSH-RC, the performance loss is higher than 20% for the same number of requests, compared to MSH and CNFSH-RC, respectively. This is due to the fact that the RR schemes favor network slices with a low number of NFs. Therefore, the delay requirements of a few NFs result in a lower number of rejected slice requests compared to schemes without RR.

Another cost of delay-awareness is the sorting of computing nodes for almost each slice. However, the complexity of $O(|N|\log|N|)$ for a small group of nodes, which is 6 in our experiments, is negligible.

Apart from the discussion on delay-aware schemes, we have not focused on the effects of the bandwidth constraint in performance evaluations. A diverse range of design parameters for bandwidth, as well as delay and CPU constraints, can be experimented with in order to see how the proposed schemes are affected in terms of their performance. We leave this issue for a future study.

Moreover, our methodology is based on a few assumptions. First, the list of slice requests is static and never changes during runtime. Experiments with online slice scheduling and additional analysis are on our roadmap. Another point of discussion is that adopting a sorting method that takes $O(|S|\log|S|)$ time may reduce the scheduler's performance as the number of slice requests increases. However, with the longest slice request lists we used, D-CNFSH-RCRR completes its execution in roughly 17.9 ms, as seen in Fig. 3.4, offering a more than 29% increase in slice satisfaction numbers than D-NoShare-RR, which is executed in 2 ms, corresponding to a 795% increase in execution time. Meanwhile, D-CNFSH-RC achieves an 18% higher slice satisfaction number compared to D-NoShare, with a 504% increase in execution time. Fig. 3.4 shows the logarithmic scale of execution times of delay-aware schemes on the y-axis.

Lastly, in D-CNFSH, we have accelerated the code base of CNFSH [38], with in-memory allocations, and achieved around a 97% decrease in execution time. As seen in Fig. 3.4, CNFSH-RCRR completes its execution in 603 ms, while D-CNFSH-RCRR completes the same in 17.9 ms on average.

3.4.2. Performance Evaluation of the Priority-Aware LB

In this subsection, we demonstrate how our priority-aware load balancing mechanism provides performance isolation among neighboring network slices that share the same NF. The main advantage of our proposed setting is that we use CNFs instead of VNFs, which are common in the literature. However, we also provide measurements for VNFs in case our LB is integrated into a VNF-based setting. In [34], Frangoudis et al. show that running a VM takes 11.5 seconds on average. Mesbahi et al. [50]

show that even passing the role of an active VM to an already running passive VM takes 2 seconds. Using our priority-aware LB, the latency to detect a congestion (0.105 ms [40]), start a container (1.129 seconds [34]), and then direct the necessary traffic to that container (0.105 ms [40]) takes around 1.129 seconds, as shown in Table 3.1. The LB runs on Windows 8.1 with 512 MB of RAM [40] and the container is on Ubuntu 14.10 with a 16-core Intel Xeon processor and 32 GB of RAM [34]. Apart from the low latency, our mechanism is energy-saving since there is no need to start a redundant VM or a container beforehand; new containers are triggered in case of network congestion or pod failure.

Table 3.1 shows delay calculations of our proposed LB, based on the measurements of [34,40], and this thesis, in case of congestion and NF failure scenarios for both VM and pod deployments. Table 3.1 considers the usage of priority-aware LB with both VMs and pods, so that the related measurements are based on the flow depicted on Fig. 3.2. The NF placement delay is based on the average execution time of *Algorithm 3* considering the D-CNSH-RCRR scheme, which has the highest delay among the other aforementioned schemes. For all scenarios, the extra delay for high-priority slices is 0.105 ms due to the strict prioritization mechanism of the proposed LB.

The USRs of low-priority slices, on the other hand, are directed to the waiting queue in case of congestion. Each USR waits for a timeout period in the waiting queue, and later NF placement can be repeated in case the congestion takes longer than the network slice can tolerate. In this case, the delay can take around 11.489 seconds with a VM deployment and only 1.129 seconds with a pod deployment. In the case of a VM failure or a pod failure, the delay can be around 11.492 seconds or 1.132 seconds, respectively.

Table 3.1. Delay measurements for Scaling and Fail-Over Scenarios with Priority-Aware LB.

Priority	Scenarios	Failure Detection (Worst Case)	LB USR Processing (Worst Case)	Repeat NF Placement	Running NF Instance	Total Delay
High-Priority Network Slice	VM Traffic Congestion	-	0.105 ms^{**} [40]	-	-	0.105 ms
	VM Failure	-	0.105 ms^{**} [40]	-	-	0.105 ms
	Pod Traffic Congestion	-	0.105 ms^{**} [40]	-	-	0.105 ms
	Pod Failure	-	0.105 ms^{**} [40]	-	-	0.105 ms
Low-Priority Network Slice	VM Traffic Congestion	-	0.105 ms [40]	0.008 ms	11.489 s [34]	11.489 s
	VM Failure	3.199 ms^*	0.105 ms [40]	-	11.489 s [34]	11.492 s
	Pod Traffic Congestion	-	0.105 ms [40]	0.008 ms	1.129 s [34]	1.129 s
	Pod Failure	3.199 ms^*	0.105 ms [40]	-	1.129 s [34]	1.132 s
*Including 2 ms health check interval.						
**Assuming that the high-priority slice does not exceed its SLA quota.						

3.5. Conclusions

Cloud-native solutions offer great potential to satisfy the requirements of 5G use cases. Availability, performance isolation, and delay requirements are the most important of these needs since they improve QoS and enable mission-critical applications. For these purposes, redundant resources are devoted, which causes underutilized resources. In this work, we have built an availability model, a bandwidth-sharing model, a delay model, an NF sharing model, and a number of heuristic algorithms to enable the placement of slices in polynomial time, each with different features with the ability to combine them to make use of their distinct advantages. Using CNFSH-RC, we achieve an improvement of 35% in slice request acceptance rate and 28% less resource usage per request compared to the NoShare scheme and 5% improvements in slice satisfaction rate and resource consumption when compared to a previous sharing scheme [7]. Moreover, we introduce a performance isolation mechanism for shared NFs using our priority-aware LB. In addition, we propose a novel metric, the ratio of underutilized to allocated resources, which demonstrates the effect of sharing mechanisms in reducing underutilized resources. Using D-CNFSH, the ratio of underutilized to allocated resources reaches around 0.4%, while this ratio is around 30% for NoShare schemes. We conducted our experiments with 10 core and 2 RAN NFs. Hence, to our knowledge, D-CNFSH is the first NF sharing mechanism that targets 12 realistic 5G NFs. Consequently, CNFSH and D-CNFSH are designed as a reliable and scalable solution for 5G and beyond NF sharing, relying on the LB we propose and the capabilities of cloud-native environments. We encourage the following research topics in order to further prove the applicability of CNFSH and D-CNFSH and improve their performance:

- a) A thorough investigation of the effects and scalability options of NF sharing in dynamic environments.
- b) Investigation of delay, bandwidth, and other parameters on the performance of NF sharing.
- c) The use of machine learning methods for prediction of network slice requests and network behaviour.
- d) Investigation of the security implications of NF sharing.
- e) Real-life applications of 5G and beyond use cases that use shared NFs.

4. SLICESCORE: A SLICE ORIENTED DDoS FILTERING APPROACH

4.1. Introduction

The idea of NF sharing has been brought up as a way to make use of underutilized resources and effectively reduce costs. Despite the fact that there are many optimization opportunities with NF sharing, there are also new security risks [35]. Due to NF sharing, the network is exposed to additional attack surfaces and possible threats. To ensure the reliability of next-generation communication networks, these problems must be resolved. Appropriate defense mechanisms are required to protect shared NFs from malicious activities that exploit these vulnerabilities.

The threat of DDoS attacks is one of the major security risks related to NF sharing. These attacks have the potential to seriously harm network performance and violate dependability. It is essential to create a secure architecture that can recognize and neutralize various threats in order to handle DDoS attacks.

In this thesis, we examine the security implications of NF sharing in next-generation networks. We identify new attack vectors that result from the sharing of NFs and provide some of the defense techniques that mitigate these risks. In addition, we provide SliceScore, a network slice-oriented DDoS filtering approach that aims to improve the security of shared NFs and network slices as well. Unlike earlier filtering techniques that ignored taking into consideration the variations across network slices, SliceScore intends to make use of the unique characteristics of each network slice to efficiently identify and prevent attackers.

We show through network simulations that SliceScore performs better than current DDoS filtering techniques, offering a reliable solution for NF sharing security in next-generation networks. Our results underscore how crucial customized security

measures are to protect shared resources and ensure dependable operation of next-generation networks. Our contributions in this thesis are the following: We first list the attack vectors that arise solely from NF sharing; second, we propose SliceScore, which incorporates the characteristics of network slices into the threshold calculation for more effective DDoS filtering compared to a previous high-performing solution, SFC [1].

The remaining sections are organized as follows. Sec. 4.2 describes the new attack vectors introduced by NF sharing. In Sec. 4.3, we introduce SFC [1], a proactive and collaborative DDoS filtering mechanism. Sec. 4.4 explains SliceScore in detail. In Sec. 4.5, we evaluate the performance of SliceScore through simulations and compare it with the benchmark. Finally, Sec. 4.6 concludes the chapter and shows potential directions for future research in the domain.

4.2. The Attack Vectors

NF sharing, while offering significant advantages in terms of resource optimization and cost reduction, also introduces new security challenges. This subsection explores the new attack vectors that arise from NF sharing, highlighting the increased risk and complexity of securing such environments.

The attack vectors described in this section are categorized into two classes:

- A) Attacks through the compromise of a network slice
- B) Attacks through the compromise of the physical or virtual infrastructure, i.e., the physical node or the NF

All attack types are also valid for environments without NF sharing; however, the effect of attacks can increase dramatically when NF sharing is used. Moreover, the attack vectors themselves might overlap as well in certain cases; however, we prefer to explain them separately to discuss and emphasize their severity. Table 4.1 summarizes

the studies in the literature that mention these attack vectors. In fact, most of these studies focus on the security of network slicing rather than NF sharing. In this thesis, we elaborate on the security risks and their implications that arise from NF sharing.

4.2.1. Cross-Slice Attacks

In an environment that allows NF sharing, the resources provided by an NF are simultaneously exposed to multiple network slices. This shared exposure inherently increases the security risks associated with confidentiality, integrity, and availability. Since all slices relying on a shared NF operate within the same software framework, any vulnerabilities in slice isolation mechanisms or shared resources can be exploited.

One of the primary risks in such a setting is unauthorized access to sensitive data. If isolation mechanisms are not strictly enforced, an attacker who compromises one slice may gain visibility into another slice's data. This can lead to data leakage, where confidential information is exposed across slices, violating the privacy of users and organizations relying on those slices [51]. Such a breach is particularly dangerous in multi-tenant environments, where slices are often operated by different service providers with competing or highly regulated interests.

Another concern is the potential for disruption in service availability. Since slices share computational and network resources, a compromised slice can be used to initiate excessive resource consumption, leading to degraded performance or even service outages for other slices. This type of disruption can be particularly damaging in critical applications such as emergency communication networks or industrial IoT deployments, where even short periods of downtime can have severe consequences. Further details on such attacks are given in Subsection 4.2.2.

Furthermore, if an attacker gains control over a slice, they may attempt to escalate their privileges within the shared NF environment (Subsection 4.2.5). This could allow unauthorized modifications to NF configurations or manipulation of network traffic.

Even when strict access controls are in place, the inherent complexity of NF sharing introduces risks related to shared memory, processing power, and inter-slice communication. In some cases, attackers may exploit side-channel vulnerabilities, leveraging shared hardware resources to infer or extract sensitive information from other slices without direct access.

To mitigate these risks, strong authentication and authorization mechanisms must be implemented to prevent unauthorized interactions between slices. Additionally, ensuring strict isolation at the software, virtualization, and network levels is crucial to maintaining security. However, cross-slice attacks remain a significant concern because they often serve as an entry point for more complex and damaging security breaches. Once an attacker gains a foothold in one slice, they may leverage their position to launch broader attacks against other slices or even the underlying infrastructure, making cross-slice vulnerabilities a critical issue in NF-sharing environments.

4.2.2. DDoS Attacks

DDoS attacks remain one of the most significant threats to modern communication networks, including 5G and beyond. By overwhelming network resources with a massive volume of malicious traffic, DDoS attacks disrupt service availability and severely degrade the performance of NFs [19, 52]. In NF-sharing environments, the impact of such attacks is particularly severe, as compromised slice isolation can lead to cascading failures that affect multiple services hosted on shared infrastructure.

The growing adoption of virtualization and containerization in 5G networks introduces new vulnerabilities, as attackers can exploit shared resources and interdependencies between slices. Network slices are designed to provide tailored services with specific QoS requirements, yet maintaining performance under DDoS conditions remains a major challenge. If an attacker successfully launches a DDoS attack against a shared NF, the excessive traffic can consume CPU cycles, memory, or bandwidth,

preventing legitimate traffic from being processed effectively. This can cause not only service disruptions for a single slice but also widespread network instability.

A particularly concerning scenario arises when a DDoS attack specifically targets security modules within an NF, such as authentication systems, firewall mechanisms, or intrusion detection components. If these security functions are overwhelmed, it can leave the NF and the entire network more vulnerable to secondary exploits, enabling attackers to bypass defenses and escalate their access. Such attacks can result in breaches of SLAs, leading to financial losses, regulatory penalties, and reputational damage for network operators and infrastructure providers [53].

DDoS attacks in NF-sharing environments can originate from multiple sources: a compromised network slice, a vulnerability within a shared NF, or even a direct assault on the underlying physical infrastructure. Since these attacks exploit both vulnerabilities in both network slices and the underlying infrastructure, they are relevant to both Class A and Class B attack vectors. However, this thesis primarily focuses on mitigating DDoS attacks that originate from Class A attacks, where a compromised slice is leveraged to launch attacks against shared NFs.

Several studies have explored potential solutions for mitigating DDoS attacks in network slicing environments. Javadpour et al. [54] propose a Slice Isolation-Based Reinforcement Learning scheme to counter DDoS attacks targeting network slices. Similarly, Sattar and Matrawi [55] introduce an on-demand slice isolation approach to mitigate DDoS attacks while ensuring end-to-end delay guarantees in 5G core networks. Additionally, Kuadey et al. [52] utilize a long-term memory deep learning technique to classify User Equipment (UE) traffic and distinguish DDoS activity from legitimate usage. These approaches highlight the importance of proactive detection, efficient mitigation strategies, and robust slice isolation mechanisms in securing NF-sharing environments against DDoS threats. We explain the limitations of the solutions mentioned above in Subsection 4.2.7.

Table 4.1. Attack Vectors Against NF Sharing Mentioned in the Literature.

Reference	Attack Vectors	Class	NF Sharing Aware	Solutions							Limitation	Legend
				Deep Learning	Blockchain	Slice Isolation	Trust/Security Zones	Zero-Trust	Reactive Configuration	Slice-Aware Filtering		
Zharabad <i>et al.</i> [6]	1, 5	A	✓	✗	✗	✓	✗	✗	✗	✗	Hardware Isolation	1: Cross-Slice Attacks
Kuadey <i>et al.</i> [52]	2	A	✗	✓	✗	✗	✗	✗	✗	✗	Training Data	2: DDoS
Hewa <i>et al.</i> [56]	2	A-B	✗	✗	✓	✗	✗	✗	✗	✗	Computation	3: Side-Channel Attacks
Thantharate <i>et al.</i> [57]	1, 2, 5	A	✓	✓	✗	✗	✓	✗	✗	✗	Training Data	4: Exploiting Missing Security
Sattar and Matrawi [55]	2	A-B	✓	✗	✗	✓	✗	✗	✗	✗	Hardware Isolation	Zones
Schinianakis <i>et al.</i> [53]	1, 2, 4-6	A-B	✗	✗	✗	✗	✓	✗	✗	✗	No DDoS Prevention	5: Elevation of Privilege
Li <i>et al.</i> [58]	3	A-B	✓	✗	✗	✓	✗	✗	✗	✗	SCA-aware only	Attacks
Kotulski <i>et al.</i> [59]	1-6	A-B	✓	✗	✗	✓	✗	✗	✗	✗	Hardware Isolation	6: Exploiting Configuration
Chen <i>et al.</i> [60]	1, 2, 5	A-B	✗	✗	✗	✗	✗	✓	✗	✗	No DDoS Prevention	Mistakes
Wang <i>et al.</i> [61]	6	A	✗	✗	✗	✗	✗	✗	✓	✗	Reactive Conf. only	A: Through slice compromise
Marquez <i>et al.</i> [62]	6	A	✓	✗	✗	✗	✗	✗	✓	✗	Reactive Conf. only	B: Through infrastructure
SliceScore	2	A	✓	✗	✗	✗	✗	✗	✗	✓	None of the above	compromise

4.2.3. Side-Channel Attacks

Side-channel attacks (SCAs) exploit unintended information leakage through shared resources, allowing attackers to infer sensitive data without direct system access. Unlike conventional cyberattacks that target software vulnerabilities or cryptographic weaknesses, SCAs take advantage of physical and behavioral characteristics of computing systems, such as variations in execution time, power consumption, electromagnetic emissions, and cache access patterns [58, 63]. These attacks can be particularly dangerous in multi-tenant environments, where different network slices or services operate on the same physical or virtualized infrastructure. Since SCAs do not rely on injecting malicious code or directly interacting with protected data, they are often difficult to detect using traditional security measures.

In the context of NF sharing, SCAs pose a critical threat due to the close proximity of multiple network slices operating on shared hardware and software resources. When an attacker gains access to a compromised or less-secure slice, they can analyze low-level system behavior to extract confidential information from co-located slices. For example, timing-based SCAs can be used to infer cryptographic keys, while cache-based attacks may reveal data access patterns that expose the nature of network traffic or security mechanisms in place [64, 65]. The increasing reliance on virtualization and containerization in CNFs further exacerbates these risks, as shared execution environments introduce additional attack surfaces. As a result, SCAs are a growing concern for NF sharing architectures, requiring advanced isolation and mitigation strategies to protect sensitive operations.

One of the primary concerns with SCAs is their ability to compromise the confidentiality of cryptographic operations. By analyzing runtime activity data, an attacker may deduce encryption keys, allowing them to intercept or manipulate protected communications [66, 67]. Additionally, SCAs can reveal patterns in network traffic, which may expose sensitive operational details such as user behavior, application usage, or security mechanisms in place. The presence of shared execution environments in NF

sharing scenarios makes it easier for adversaries to gather such information without triggering traditional security alarms.

Mitigating SCAs in NF sharing environments requires a multi-layered approach. Robust isolation mechanisms should be implemented to minimize shared resource interference, ensuring that slices cannot inadvertently reveal critical information. Secure cryptographic designs must incorporate countermeasures such as constant-time execution and randomization techniques to obscure patterns that attackers could exploit [68]. Additionally, continuous monitoring tools should be deployed to detect abnormal resource usage that may indicate an ongoing SCA [69]. Striking a balance between resource efficiency and security remains a challenge, but it is essential to safeguard the integrity and confidentiality of shared NFs [70].

4.2.4. Exploiting Missing Security Zones

The adoption of NF sharing introduces significant challenges in ensuring trust and isolation between multiple network slices. Unlike traditional 4G networks, where the core network operates within a single, well-defined security zone, 5G introduces network slicing, which requires each slice to be treated as an independent trust zone with strict isolation measures [35]. However, the coexistence of multiple tenants within shared NFs increases the attack surface, as any misconfiguration, weak security enforcement, or missing isolation controls can be exploited by malicious actors. Inadequate separation between slices allows attackers to access shared resources, manipulate traffic, or even eavesdrop on sensitive operations, threatening the integrity and confidentiality of network services [19, 59].

A key risk in NF sharing environments is the exploitation of vulnerabilities within shared NFs, where a compromised or malicious tenant could leverage weak isolation mechanisms to affect other slices. This could involve manipulating control plane communications, injecting malicious traffic, or interfering with critical NF operations. In some cases, attackers may exploit improperly configured access controls to escalate

privileges and gain unauthorized access to sensitive network data. The risk is further exacerbated by the complexity of cloud-native deployments, where microservices and containerized NFs rely on shared virtualized infrastructure, making it difficult to enforce strict security perimeters.

Moreover, security breaches stemming from missing or weakly enforced trust zones can facilitate lateral movement within the network, allowing attackers to propagate their influence beyond a single slice. Once inside, an attacker could exploit inter-slice communication channels, access shared memory, or manipulate orchestration mechanisms to disrupt service availability. Misconfigurations in authentication, policy enforcement, and resource allocation frameworks are common entry points for such attacks, highlighting the necessity of comprehensive security policies.

To mitigate these risks, strong isolation mechanisms must be enforced between network slices and tenants. Techniques such as slice-aware security policies like Slice-Score [71], which is proposed in this thesis, zero-trust architectures [72, 73], and strict access controls should be implemented to prevent unauthorized interactions between slices. Additionally, continuous security monitoring, anomaly detection, and automated response mechanisms are crucial for identifying and mitigating potential breaches before they escalate. The integrity of shared NFs must be safeguarded through hardened security configurations, robust identity management, and rigorous validation of slice-specific security policies. By ensuring well-defined security zones and preventing cross-slice interference, network operators can enhance the overall resilience of NF sharing environments.

4.2.5. Elevation of Privilege Attacks

Elevation of privilege attacks occur when an attacker, who is already an authenticated user within the system, escalates their access rights beyond what was originally intended. In the context of NF sharing, this means an attacker can gain unauthorized control over neighboring slices, shared NFs, or even the underlying infrastructure [36].

These attacks often exploit software vulnerabilities, weak authentication mechanisms, or misconfigurations in access control policies. Unlike external attacks, which rely on breaching initial defenses, elevation of privilege attacks leverage the attacker's legitimate presence within the system, making them particularly difficult to detect and mitigate.

Once an attacker gains elevated privileges, they can execute critical commands, access restricted data, modify NF configurations, or disrupt network operations. Such attacks can lead to severe consequences, including service outages, data breaches, and loss of network integrity. For instance, an attacker with administrative privileges could reconfigure routing policies, manipulate resource allocation mechanisms, or disable security controls, causing widespread disruptions across multiple tenants. Additionally, if privilege escalation occurs at the orchestration or control plane level, attackers may gain the ability to create, modify, or delete entire network slices, severely compromising service availability and security.

To mitigate elevation of privilege attacks, strict access control policies must be enforced, ensuring that users and processes operate under the principle of least privilege. Role-based and attribute-based access control mechanisms should be implemented to limit the scope of user permissions [74]. Additionally, continuous monitoring, logging, and anomaly detection systems should be deployed to identify suspicious access patterns and privilege escalation attempts in real-time. Regular security audits, vulnerability assessments, and automated patching of known exploits are also essential in preventing privilege escalation within shared NF environments.

4.2.6. Exploiting Configuration Mistakes

Despite the layered security architecture of 5G core networks, the complexity of security configurations introduces significant risks. Modern networks rely on precise settings for access control, encryption, traffic routing, and resource allocation. However, even minor misconfigurations in these areas can create severe vulnerabilities. A

single error—such as an overly permissive firewall rule, an incorrectly defined trust relationship, or a misapplied security policy—can inadvertently expose sensitive data, enable unauthorized access, or weaken network segmentation [59,60]. Attackers actively scan for such weaknesses, leveraging them as entry points to bypass otherwise robust security defenses.

Configuration errors can stem from both manual and automated processes. Manual configurations, especially in large-scale and dynamic environments like 5G, are prone to human mistakes due to the complexity and interdependence of network settings. Automated configuration management tools help reduce human errors but are not immune to flaws—misaligned policies, improper testing, or software bugs can propagate vulnerabilities at scale. The increasing adoption of network slicing and NF sharing further complicates configuration management, as each slice may require distinct security policies that must be consistently enforced across multiple layers of the network.

Mitigating risks associated with configuration mistakes requires a multi-layered approach. Automated validation tools should be employed to detect misconfigurations before deployment, and real-time monitoring mechanisms should continuously verify compliance with security policies. Additionally, regular security audits, penetration testing, and adherence to standardized configuration guidelines can help prevent misconfiguration-related exploits. Continuous training for network administrators and security teams is also critical to maintaining a high level of expertise in managing evolving security requirements.

4.2.7. Discussion on Previous Solutions

As shown in Table 4.1, most of the references propose slice isolation as a solution to the listed vulnerabilities. Sattar and Matrawi [55] propose hardware isolation among network slices. According to [51], [59], NF sharing is contrary to slice isolation, and there should be further security improvements to be able to benefit from NF sharing in

a secure way. Zharabad et al. [6] claim that hardware isolation is the most beneficial way to secure slices. In contrast, Li *et al.* [58] propose a resource allocation algorithm that is SCA-aware in which virtual isolation among slices is considered.

Deep learning techniques require a large amount of training data, which is challenging in the networking domain. Thantharate et al. [57] use a 65K dataset for their Secure5G solution. Meanwhile, blockchain models are effective but have computational overload as shown by [75]. Trust/Security Zones and Zero-Trust techniques define applicable frameworks for a wide range of scenarios; however, they do not include a DDoS prevention mechanism.

The majority of the aforementioned techniques concentrate on several attack vectors, while DDoS is the common threat among them. Although some of the authors in Table 4.1 are aware of the security challenges associated with NF sharing, others focus only on the security of slices that use dedicated NFs. In this thesis, we show how isolation can be achieved despite using NF sharing by thoroughly evaluating the slice attributes. The slice-oriented DDoS filtering mechanism we propose offers a fine-grained approach to secure NF sharing environments.

4.3. Previous Work in DDoS Filtering: ScoreForCore

DDoS attacks can cause high financial and reputational damage to institutions. In order to tackle this problem and contribute to the literature, we have worked on various DDoS filtering mechanisms [1, 76].

In [1], we proposed SFC, which enables cooperation among core network routers in order to achieve the most effective tuple of packet attributes that characterizes the DDoS attack. SFC calculates the score of each packet based on traffic characteristics before and during the attack. SFC is called a proactive filtering mechanism since it starts processing before the attack starts, as shown in Table 4.2. The threshold is calculated for the entire network so that the ratio of allowed traffic to actual traffic is

considered. The score of each packet is compared to the threshold, and if it is larger than the threshold, it is dropped.

As a result, SFC achieves higher percentages for the metrics used compared to a previous study [77], because SFC considers dynamic detection of effective attributes in the DDoS attack.

In this thesis, SliceScore outperforms conventional filtering algorithms that do not take into account the variations among slices by utilizing the distinctive qualities of network slices. SliceScore improves the general security of shared NFs by incorporating a slice-oriented traffic filtering technique, guaranteeing the reliable functioning of next-generation networks.

The architecture and functionality of SliceScore are covered in detail in Sec. 4.4. Comprehensive simulations and comparisons with another filtering mechanism are made to show its effectiveness in Sec. 4.5. Our efforts demonstrate the value of adaptive security techniques and offer an advancement in the security of NF sharing and next-generation communication networks.

4.4. Design and Operation of SliceScore

The simplicity in running DDoS attacks and their wide range of effects compared to other types of attacks make NF sharing and network slicing prone to resource exhaustion risks. To address this threat, we focus on the mitigation of DDoS attacks and propose SliceScore, which integrates the characteristics of network slices into an existing DDoS filtering mechanism, SFC [1].

In SFC, all network traffic, regardless of where the packets are aimed, is treated in the same way. However, in a communication network where network slicing is used, each network slice can be treated with a fine granularity so that a thorough diagnosis and mitigation of DDoS can be achieved. A slice with less bandwidth size compared

to others can be attacked with a DDoS traffic density that is negligible for other slices. Therefore, detection and mitigation of such DDoS attacks would be impossible without considering the slice attributes.

SliceScore is designed to improve the security of NF sharing by incorporating network slice parameters into the calculation of thresholds and packet scores. This approach ensures that the unique characteristics and requirements of each network slice are taken into account, providing a more precise and effective DDoS mitigation mechanism.

In order to mitigate a DDoS attack, we divide the network lifetime into two phases: the attack-free phase and the attack phase. In the attack-free phase, the bandwidth usage is below the capacity of links used, where we assume the network is free from DDoS attacks. Therefore, in this phase, nominal profiles that include the nominal values for the attributes of network slices are collected. The attributes we consider consist of source IP, protocol type, packet size, TTL value, TCP flag, destination IP, and destination port. Due to this preparation phase before the attack starts, SliceScore is also a proactive mechanism.

Table 4.2. Comparison of DDoS Filtering Techniques.

Reference	Proactive	Collaboration	Iceberg Profiles	Scoring/Metrics	Threshold
HCF [78]	✗	✗	✗	Static Attributes	Network-Wide
PacketScore [77]	✓	✗	✓	Static Attributes	Network-Wide
SSM [79]	✓	✗	✗	Static Attributes	Network-Wide
SFC [1]	✓	✓	✗	Dynamic Attributes	Network-Wide
SliceScore	✓	✓	✓	Slice-Oriented Dynamic Attributes	Slice-Specific

Table 4.2 compares SliceScore with other DDoS filtering mechanisms in terms of proactivity, collaboration, usage of iceberg profiles, scoring characteristics, and thresh-

old coverage. Jin *et al.* [78] propose Hop Count Filtering (HCF), which eliminates packets based on the accuracy of their TTL values. Kim *et al.* [77] propose PacketScore, where static attributes are used in profiling. PacketScore uses iceberg profiling, which is also adopted into SliceScore. Kalkan and Alagöz [1] use dynamic attributes and collaboration among routers to decide on suspicious packet features in SFC. This comparison demonstrates that SliceScore, with its slice-oriented mechanism, not only introduces a proactive approach but also provides granular and dynamic DDoS mitigation in one place that traditional methods lack.

The attack phase starts when the network monitoring system detects a large volume of traffic beyond previously defined limits and gives a DDoS attack alert. This phase is divided into periods of equal length. The current period is represented by t , starting with $t = 0$. In this phase, the attack profiles are constructed based on the number of packets that have the same values for each attribute and for each *OwnPair*^s, which is explained below and in Fig. 4.1.

For memory-saving purposes, two randomly chosen attributes for each slice are called the *OwnPair* of the slice and are represented by *OwnPair*^s. The OwnPair Nominal Profile for slice s , *OPNP*^s, holds the number of packets for the values of *OwnPair*^s.

SFC [1] relies on the cooperation of distributed routers to achieve an effective attribute tuple with low memory consumption. In addition to that, in SliceScore, we make use of the iceberg-style profiles that are described in [77] in detail. In an iceberg-style profile with a static threshold, the attribute values that appear less than the threshold in the network are not stored. Instead, the static threshold is used when needed. As a result, we further reduce the amount of memory used to count the occurrences of attribute tuples in SliceScore.

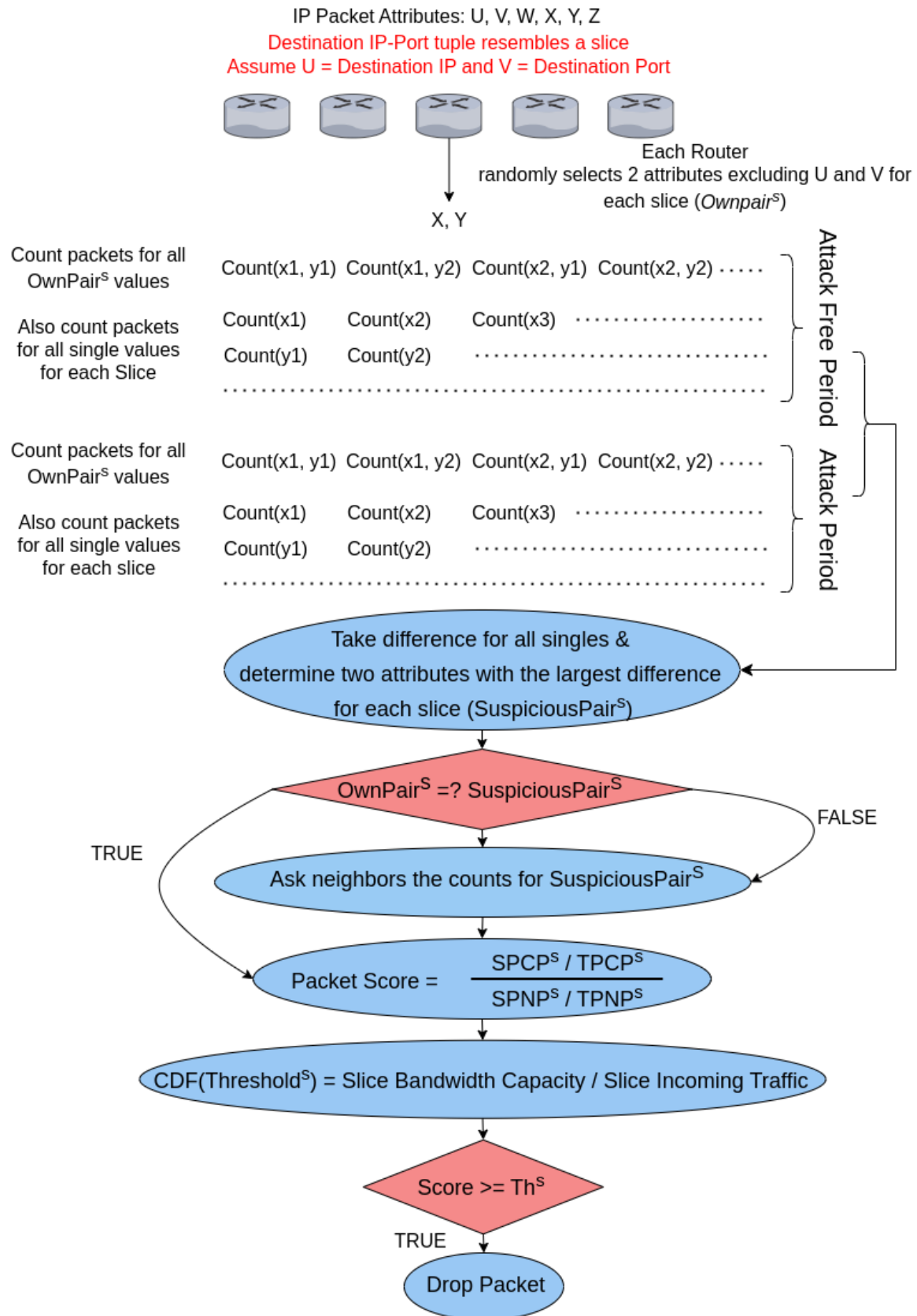


Figure 4.1. SliceScore.

Each slice s is a member of the set of slices S . For each packet attribute other than destination IP and port, the number of packets targeting the slice s in the attack-free phase that have the same attribute values is recorded, and this record is called the Single Nominal Profile of s (SNP^s). The same is maintained for the attack phase and is called the Single Current Profile of s (SCP^s).

The two attributes that have the most deviating SCP^s compared to their SNP^s based on each slice s are determined and called *SuspiciousPair* ^{s} . Meanwhile, the pair that should be considered for the calculation of the packet score for the same slice is *ScorePair* ^{s} , which is determined as the following: If the *OwnPair* ^{s} of a router is the same as *SuspiciousPair* ^{s} , then *OwnPair* ^{s} is assigned as *ScorePair* ^{s} . If not, the router asks its one, two, and three-hop neighbors, respectively, for the nominal profile of *SuspiciousPair* ^{s} . If it is found, then *SuspiciousPair* ^{s} is assigned to *ScorePair* ^{s} . Otherwise, *OwnPair* ^{s} is used as *ScorePair* ^{s} . The ScorePair Nominal Profile of the slice s , $SPNP^s$, and the ScorePair Current Profile of the slice s , $SPCP^s$, are set accordingly. The collaboration among routers is explained in more detail in our previous paper [1].

Starting from $t = 1$, following the initial construction of nominal and current profiles, the score of every packet is calculated and the slice thresholds are updated with each packet received. If the score of a packet is smaller than the current threshold, the packet is passed; otherwise, the packet is dropped. In this way, SliceScore aims to turn the traffic usage of each slice below bandwidth capacity, which is DDoS safe.

SliceScore is a slice-oriented DDoS filtering approach that considers the characteristics of network slices to differentiate between legal and illegitimate traffic. Traditional filtering approaches that ignore the characteristics of the network slices might treat the legal traffic of a high-bandwidth slice as a suspected attack while underestimating a real attack that is smaller in size against a slice that can handle only a small amount of bandwidth. SliceScore is designed to handle these cases as well, which is not only beneficial for DDoS filtering but can also be applied for traffic throttling.

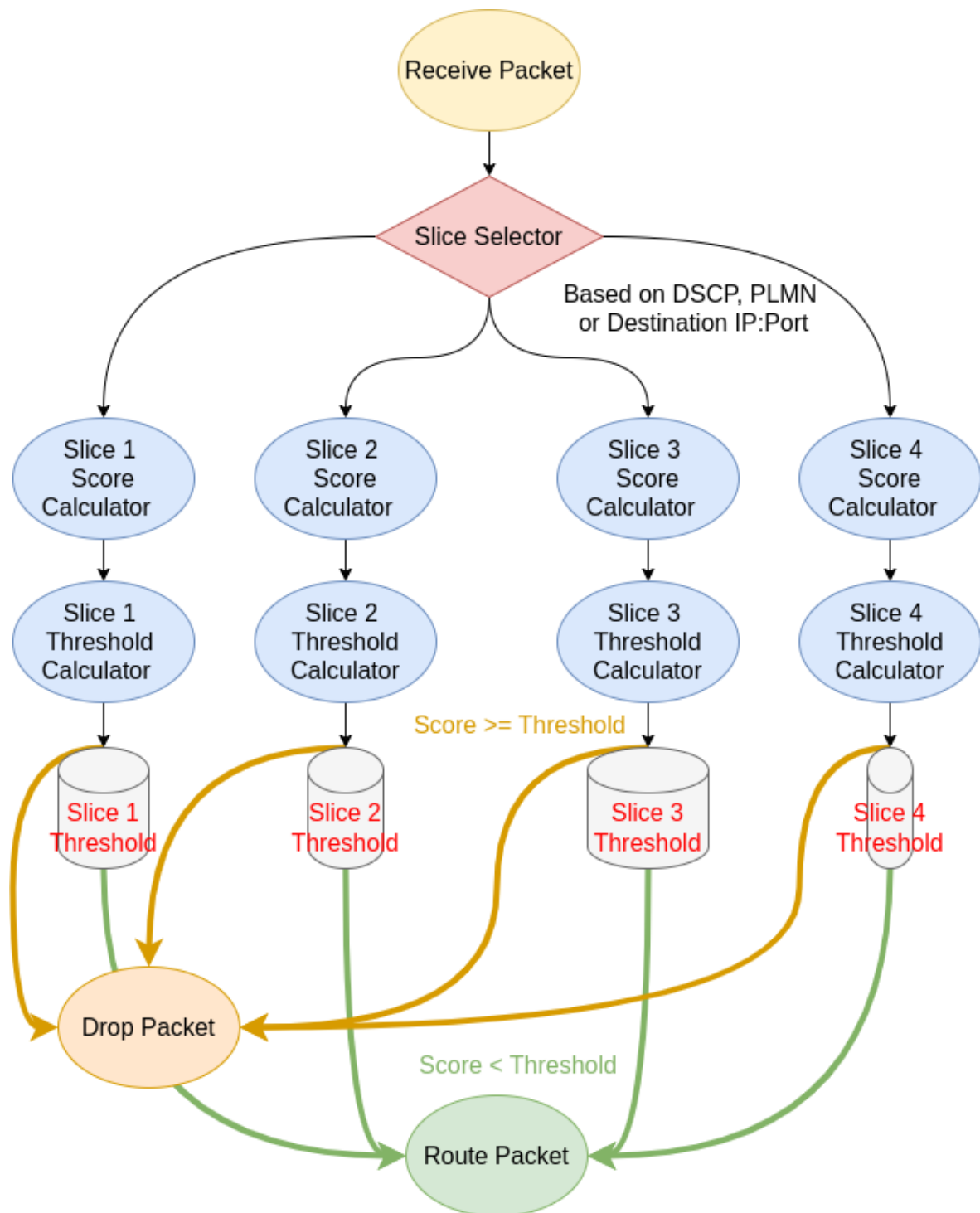


Figure 4.2. Packet Processing in SliceScore.

As shown in Fig. 4.2, the main criterion for a packet to pass the SliceScore filter is that its score should fit in the threshold for the slice it is targeting. Therefore, packets of any DDoS attack are filtered out if they do not fit the predefined bandwidth

limits of the slice. If some amount of DDoS traffic is allowed due to having some idle bandwidth capacity, then they are not supposed to have a DDoS effect, as implied by Eq. 4.1, 4.2 below.

In a 5G setting supported with NF sharing, network slices can be distinguished in three ways: with the Differentiated Services Code Point (DSCP) field of the IP packet, like in [80]; with packet attributes special to the protocol used, such as the value of the Public Land Mobile Network (PLMN) of an NGAP protocol [81]; or with the destination IP-port pair. For simplicity, we use the destination IP-port pair to differentiate a network slice. In addition, we assume that each network slice has a bandwidth capacity defined in the corresponding SLA.

The score of each incoming packet is calculated with

$$Score = \frac{SPCP^s / TPCP^s}{SPNP^s / TPNP^s}, \quad (4.1)$$

where $TPCP^s$ is the total number of packets in the current profile of s , and $TPNP^s$ is the total number of packets in the nominal profile of s .

The threshold Th^s for slice s is updated with each packet to satisfy

$$CDF(Th^s) = \frac{\text{Bandwidth Capacity of } s}{\text{Current Bandwidth Usage of } s}, \quad (4.2)$$

so that if the score of a packet destined to slice s is greater than or equal to Th^s , it is dropped. The ratio of bandwidth capacity (legitimate usage) of slice s over current bandwidth usage of s determines the ratio of allowed traffic over actual traffic. In this sense, a packet should have a score less than Th^s to be allowed.

If a packet is among the attack packets, its score is expected to be higher than that of the packets that are not related to the attack. This is achieved by scoring a packet in the following way: The ratio of occurrence of packets with the same features with that packet over all packets in the attack phase is divided by the ratio of occurrence of packets with the same features over all packets in the nominal phase (Eq. 4.1). Eq. 4.1, 4.2 can be used for various network topologies, scales, and DDoS intensities due

to the fact that SliceScore is slice-oriented, hence scalable in nature. Furthermore, in complex DDoS attack scenarios, the global view of the DDoS attack might be necessary for an effective filtering mechanism, which is one of our future research topics.

Another sophisticated attack strategy that adversaries may employ is the hit-and-wait method, where a DDoS attack is launched for a short duration, then stopped, only to be resumed after some time. This intermittent attack pattern makes detection more challenging, as traffic spikes appear occasionally rather than continuously. A potential countermeasure for such attacks is the implementation of a DDoS alarm timer, which tracks recurring attack instances over a predefined time window. If multiple attack bursts are detected within this timeframe, the system can classify the source as a persistent threat and apply stricter mitigation actions, such as temporarily blocking the attacker's traffic from the network. This mechanism could further enhance the adaptability of SliceScore by improving its resilience against stealthy and periodic DDoS attempts. Future research can focus on the implementation and evaluation of this strategy within SliceScore to assess its effectiveness in mitigating hit-and-wait attack patterns.

4.5. Performance Evaluation

The effectiveness of SliceScore was evaluated using the ns-3 network simulator. Version 3.30 of ns-3 was run in a Docker 26.1.4 container on Ubuntu 20.04.6 running on an Intel Core i7-8550U CPU @1.80GHz with 16 GB of memory. As shown in Fig. 4.3, the simulation environment includes 3 5G base stations (gNBs), each with 5 UEs attached. The gNBs are connected to a router that contains the SliceScore module. SliceScore traces all packets that are destined to 5G NFs and operates each packet based on the relevant phase described in Sec. 4.4. 4 hosts are connected to this router, each running a server listening on 2 UDP ports, 53 and 1434. Each of these ports represents a slice. The bandwidth capacities of these eight slices are assigned incrementally between 50-85 Mbps. The attack-free and attack phases are simulated, excluding the collaboration phase, which is already analyzed in our previous work [1].

In this thesis, we focus on the effects of the simulation parameters on the success of our slice-oriented approach, SliceScore.

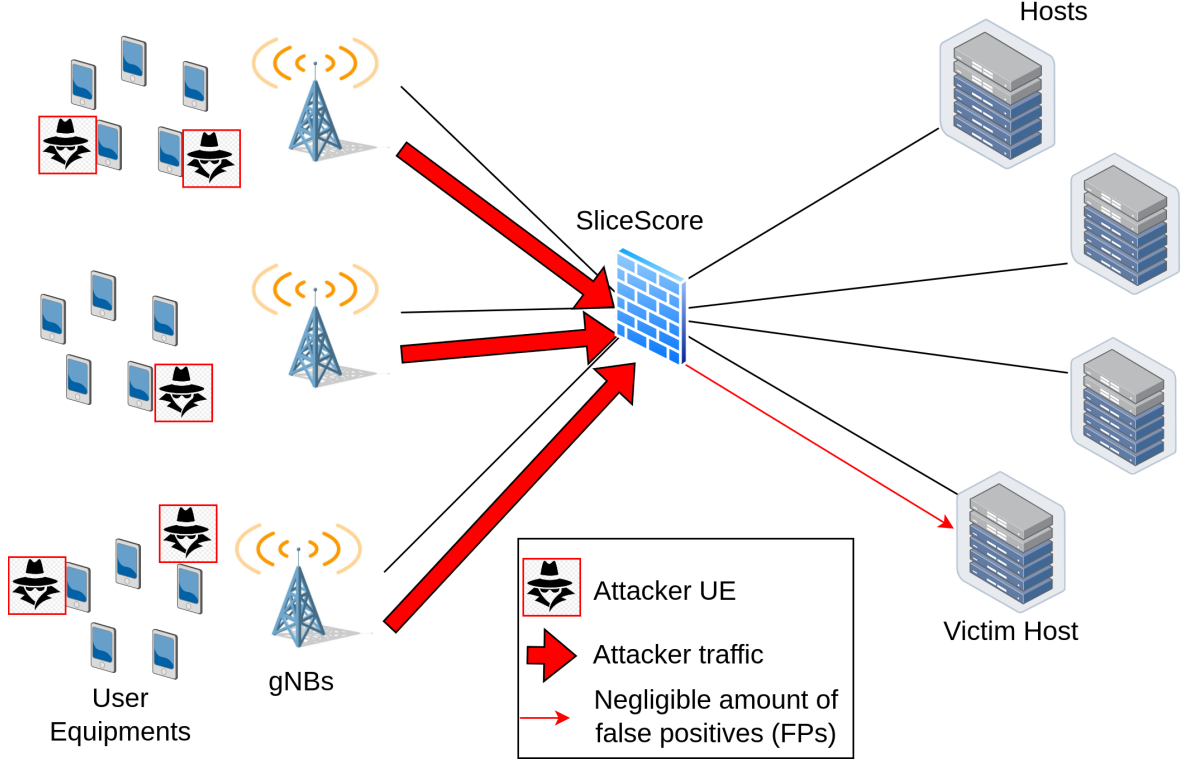


Figure 4.3. NS-3 simulation settings with specific victim scenario.

In the beginning, each UE randomly selects six slices and sends 5 Mbps of traffic to each of them until the end of the simulation. 2 seconds later, the attack phase starts with 5 randomly selected attackers among the UEs. The total attack size is 400 Mbps for the first set of simulations and 1.2 Gbps for the second set. Each scenario explained below was executed with both attack densities and ten times each for accuracy.

- Random Attack (RA): Each attacker chooses a random victim among the eight available slices with packets of random size between 40 and 1000 bytes.
- Random Attack on the slice with the least bandwidth capacity (RAL): All attackers attack the slice with 50 Mbps capacity with packets of random size between 40 and 1000 bytes.

- Random Attack on the slice with the highest bandwidth capacity (RAH): All attackers attack the slice with 85 Mbps capacity with packets of random size between 40 and 1000 bytes.
- DNS Amplification Attack on Random Victims (DNSR): Each attacker attacks a randomly selected slice with 60-byte packets.
- DNS Amplification Attack on the slice with the least bandwidth capacity (DNSL): All attackers attack the slice with 50 Mbps capacity with 60-byte packets.
- DNS Amplification Attack on the slice with the highest bandwidth capacity (DNSH): All attackers attack the slice with 85 Mbps capacity with 60-byte packets.

For comparison, we simulate the same scenarios with SFC [1], which is an example of a DDoS filtering mechanism without considering the network slice attributes. In SFC, all packets are treated in the same way. In SliceScore, the corresponding network slice of each packet is determined first based on the destination IP-port pair. Each network slice has its own threshold, and the score of each packet is calculated based on its network slice attributes. In this way, each packet is evaluated with a slice-oriented and fine-grained approach that avoids overgeneralization.

For both SliceScore and SFC, we count the number of correct drops as True Negatives (TN), incorrect drops as False Negatives (FN), correct passes as True Positives (TP), and incorrect passes as False Positives (FP). Later, key performance indicators (KPIs), namely precision (PN), recall (RL), true negative rate (TNR), negative predictive value (NPV), accuracy (AY), F-measure (FM), and F-measure complement (FMC), which are widely used in classification problems, are calculated, as shown in Table 4.3.

PN, RL, TNR, and NPV show the success ratio of correctly identifying the allowed packets, legitimate packets, illegitimate packets, and dropped packets, respectively. AY shows the success ratio among all packets. FM is the overall success in classifying

legitimate packets, deduced from the balance between PN and RL. FMC is considering both TNR and NPV.

Table 4.3. Key Performance Indicators (KPIs).

Precision	$TP / (TP + FP)$
Recall	$TP / (TP + FN)$
True Negative Rate	$TN / (TN + FP)$
Negative Predictive Value	$TN / (TN + FN)$
Accuracy	$(TP + TN) / (TP + TN + FP + FN)$
F-Measure	$2 * PN * RL / (PN + RL)$
F-Measure Complement	$2 * TNR * NPV / (TNR + NPV)$

PN, in Fig. 4.4, 4.5, shows that SliceScore is successful over SFC in RA in both attack densities and in DNSR in 400 Mbps attacks up to 19 %. Meanwhile, in RAL in 400 Mbps attacks and RAH in both attack densities, SFC has higher values than SliceScore with up to 4 %. In SliceScore, slices with unused bandwidth capacity can allow some of the DDoS traffic without loss of service continuity. For this reason, attacks on servers with higher bandwidth capacity result in higher FP values and lower PN compared to attacks on servers with nearly full bandwidth capacity.

As shown in Fig. 4.6, 4.7, SliceScore outperforms SFC in RL for all scenarios with up to 43 %. In Fig. 4.8, 4.9, SliceScore has higher TNR values than SFC with DNSR in 400 Mbps attacks with 8%, which results in equal performance for both SliceScore and SFC in 1.2 Gbps attacks. Whereas with RAH scenarios, SliceScore allows for some of the DDoS traffic due to idle bandwidth capacities, as explained for PN above. As shown in Fig. 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, SliceScore has higher NPV, AY, FM, and FMC values for all scenarios with up to 29%, 25%, 31%, and 20%, respectively.

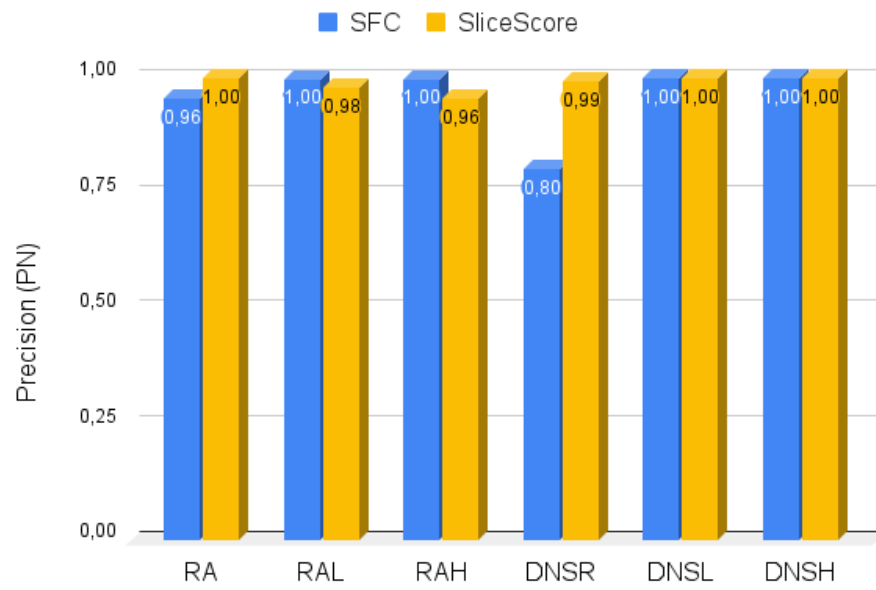


Figure 4.4. Precision in 400 Mbps Attacks.

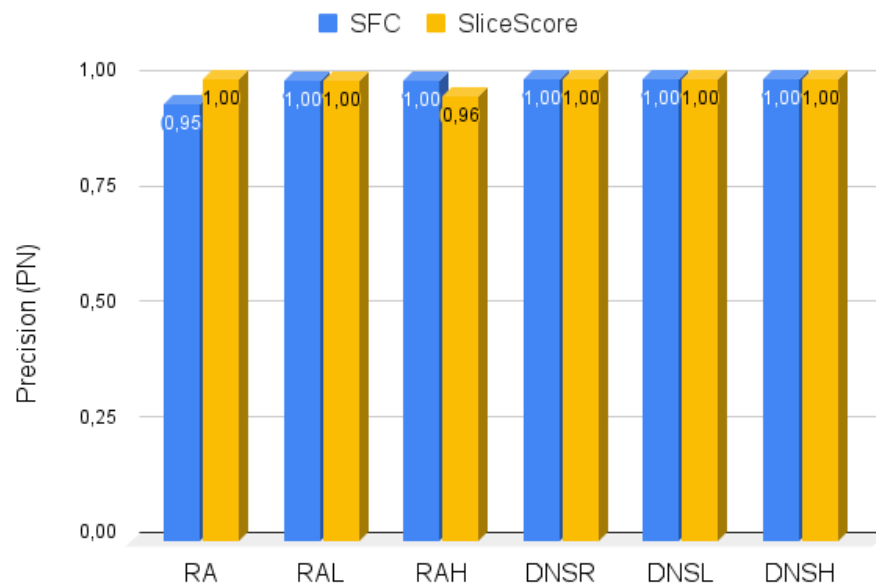


Figure 4.5. Precision in 1.2 Gbps Attacks.

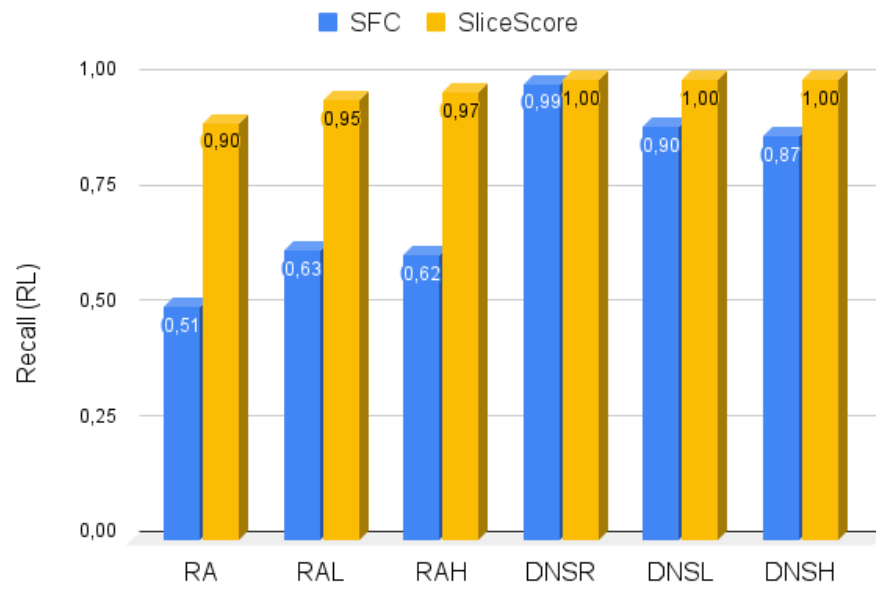


Figure 4.6. Recall in 400 Mbps Attacks.

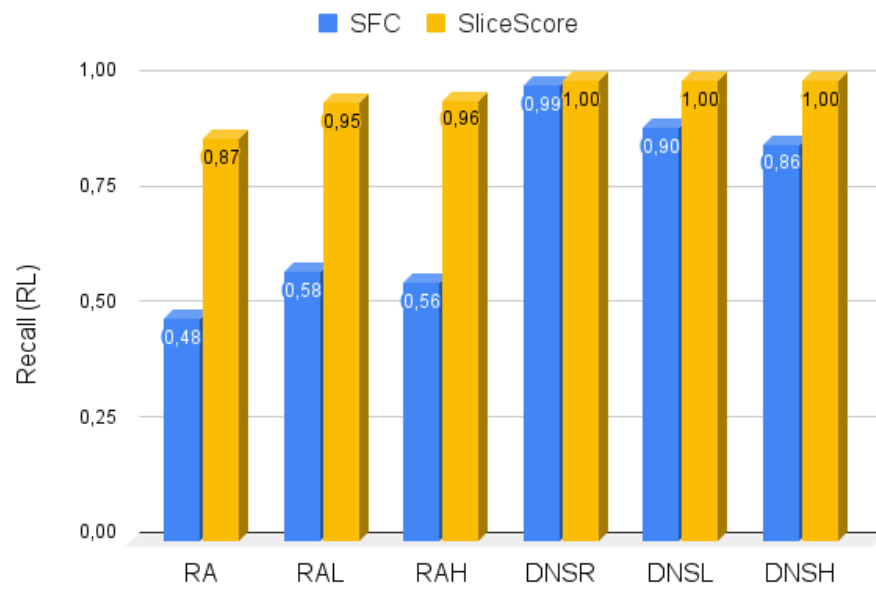


Figure 4.7. Recall in 1.2 Gbps Attacks.

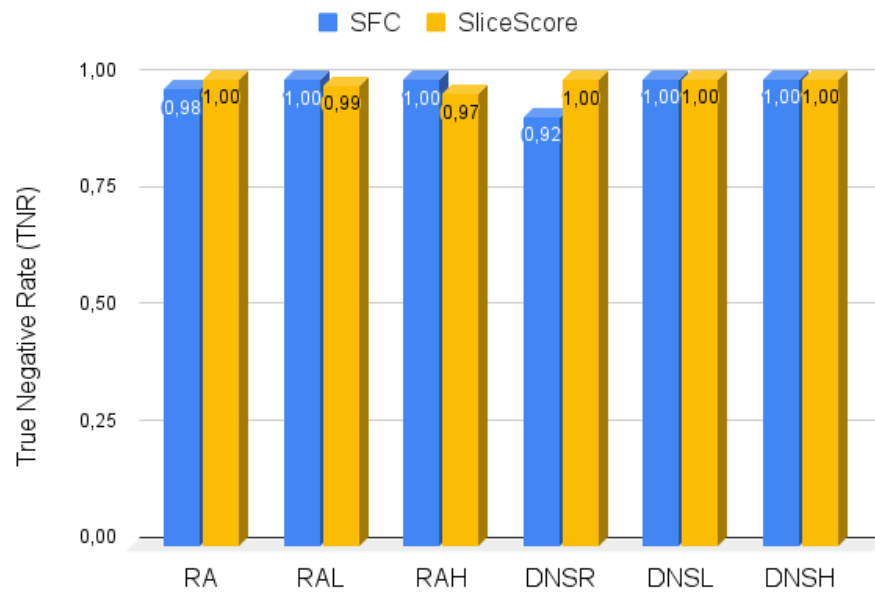


Figure 4.8. True Negative Rate in 400 Mbps Attacks.

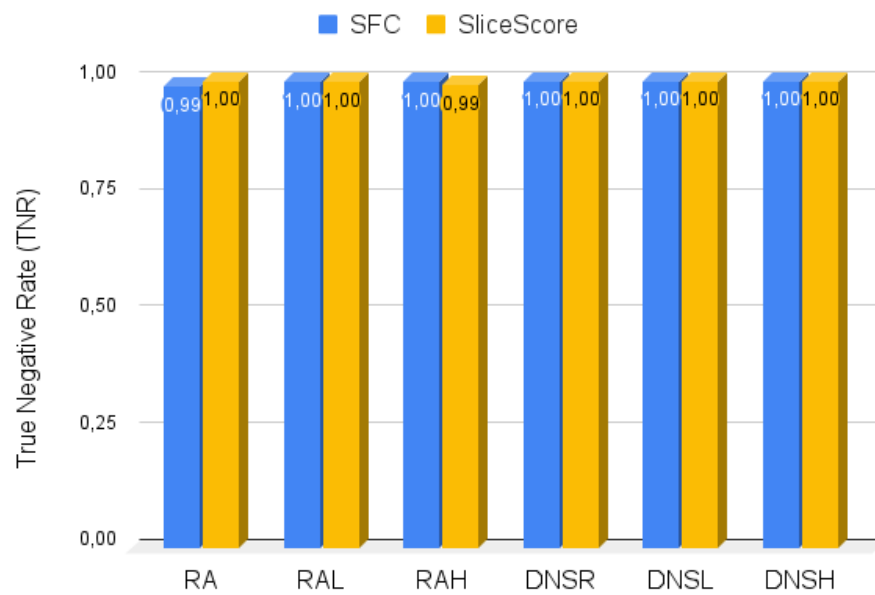


Figure 4.9. True Negative Rate in 1.2 Gbps Attacks.

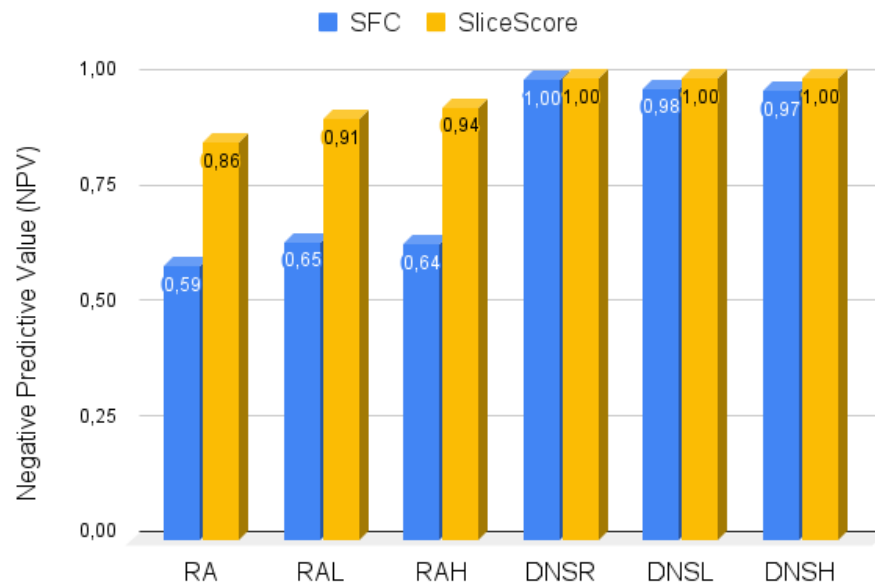


Figure 4.10. Negative Predictive Value in 400 Mbps Attacks.

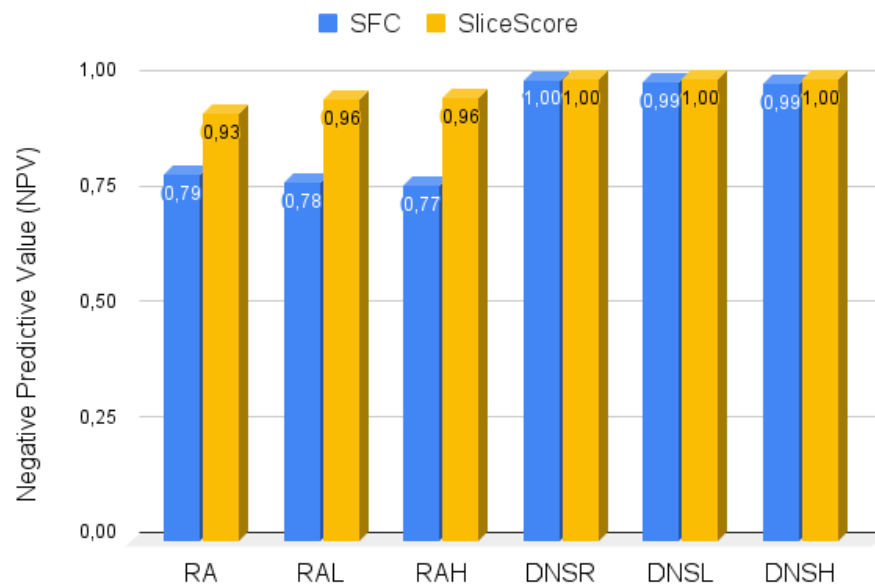


Figure 4.11. Negative Predictive Value in 1.2 Gbps Attacks.

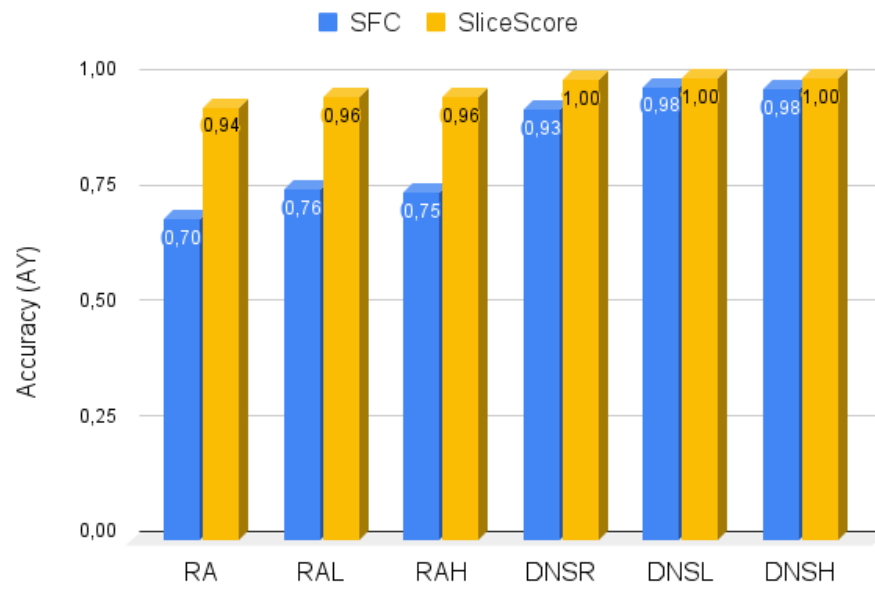


Figure 4.12. Accuracy in 400 Mbps Attacks.

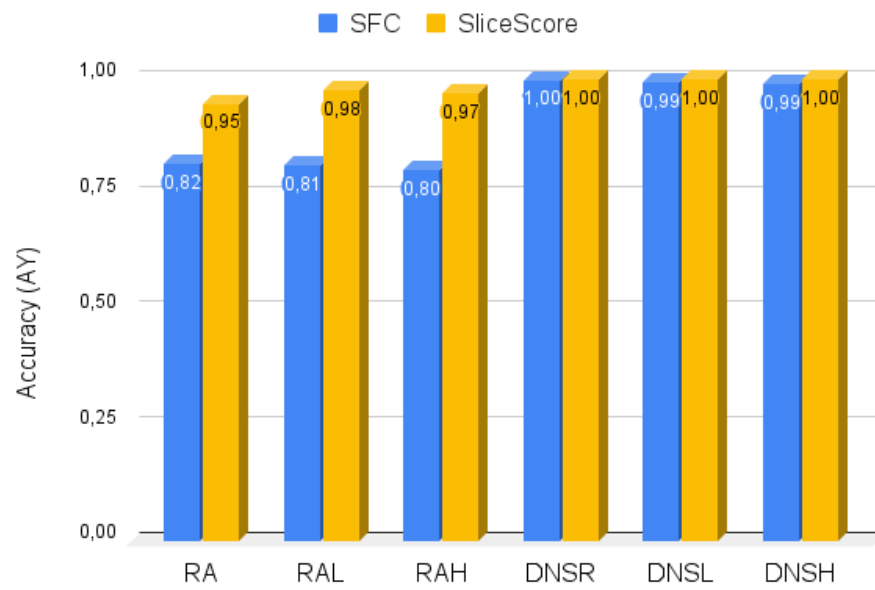


Figure 4.13. Accuracy in 1.2 Gbps Attacks.

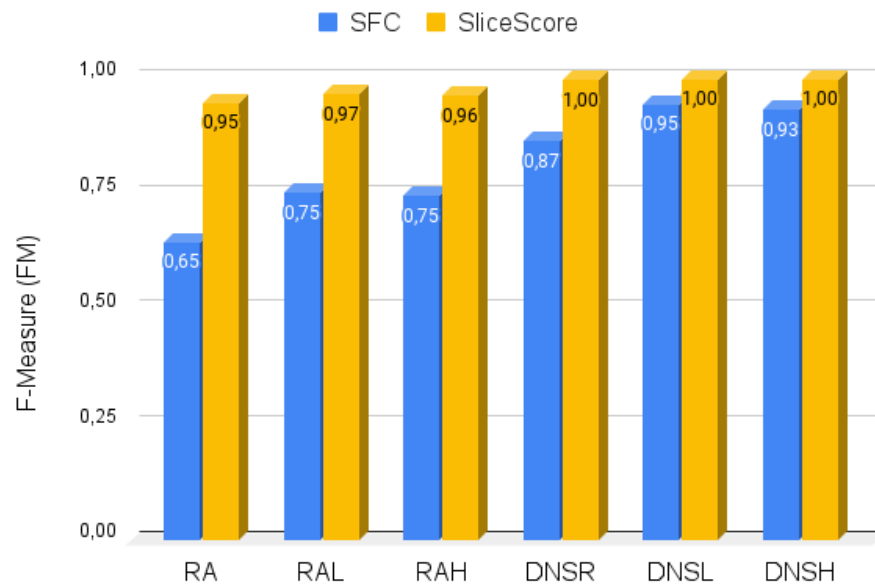


Figure 4.14. F-Measure in 400 Mbps Attacks.

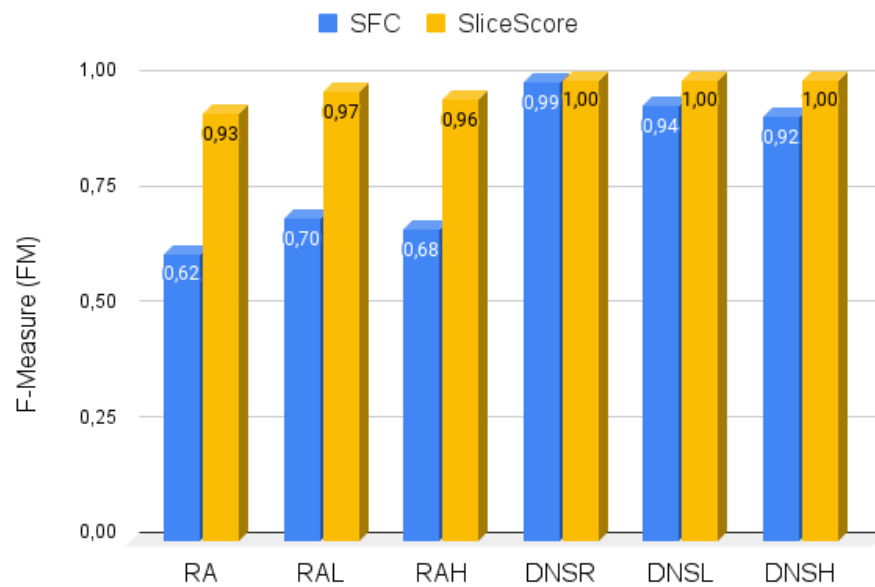


Figure 4.15. F-Measure in 1.2 Gbps Attacks.

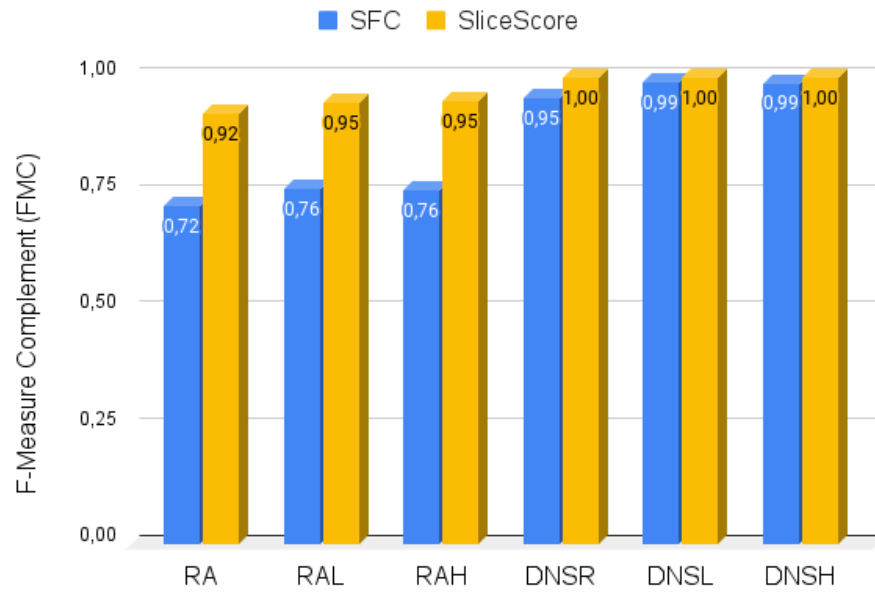


Figure 4.16. F-Measure Complement in 400 Mbps Attacks.

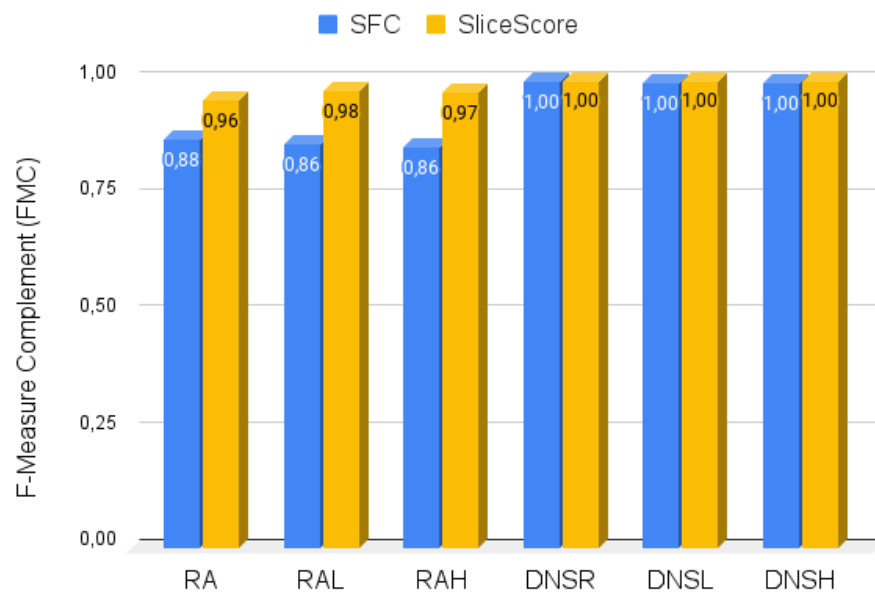


Figure 4.17. F-Measure Complement in 1.2 Gbps Attacks.

Apart from that, it is notable that in almost all scenarios, uncertainty in attack features makes DDoS mitigation harder. RA, RAL, and RAH attacks with random packet sizes include more uncertainty compared to DNS amplification attacks where

60-byte packets are used. Moreover, when the attack densities increase from 400 Mbps to 1.2 Gbps DDoS attack characteristics become more visible such that both SliceScore and SFC have more successful results with the second set of attacks.

SliceScore outperforms SFC for almost all KPIs except those considering FP. SliceScore is designed such that some amount of DDoS traffic is allowed to pass up to the limits of the slice bandwidth; however, the amount of allowed illegitimate packets is far from triggering a DDoS attack.

The main difference between SliceScore and SFC is that in the latter, all network slices are treated in the same way regardless of their bandwidth limits. As a result, including slice-specific profiles in the score calculation and the network slice bandwidth limit in the threshold calculation significantly improves detection accuracy and mitigation effectiveness for SliceScore.

4.6. Conclusions

In recent years, NF sharing has attracted the attention of researchers in order to achieve cost-effective implementations of communication networks. In this chapter, we first demonstrate how NF sharing introduces new attack vectors that target NF and slice security. Then, we present SliceScore, a DDoS filtering mechanism designed to improve the security of network slicing and NF sharing in next-generation communication networks by extending a previous work, SFC [1]. SliceScore incorporates network slice bandwidth capacity and slice-specific measurements into the threshold and score calculations, providing a more adaptive and precise approach to DDoS mitigation. Simulation results demonstrate that SliceScore outperforms traditional DDoS filtering mechanisms. This improvement highlights the importance of considering network slice characteristics in security mechanisms, ensuring more effective protection for shared resources in next-generation communication networks. Future work will focus on further expanding the slice-oriented approach to address a wider range of security threats. We also encourage further research on investigating the use of SliceScore in different

network architectures, such as edge computing and the internet of things. SliceScore is scalable in nature and can be adopted to such various topologies with slight modifications and design decisions based on the corresponding network slice features.



5. CONCLUSION

The diverse requirements of vertical use cases in 5G and beyond networks have highlighted the need for highly efficient, sustainable, and secure resource allocation strategies. One of the main challenges in this domain is the optimization of NF placement, which is essential to meet diverse service requirements while minimizing costs and resource wastage. This thesis aims to address these challenges by developing a framework that optimizes NF placement by incorporating the NF sharing concept, focusing on resource efficiency, delay sensitivity, and security concerns.

In this work, we present novel cost-efficient heuristics, CNFSH and D-CNFSH, designed specifically for 5G and beyond networks. The primary contribution was to reduce resource underutilization by maximizing shared NF usage, thus improving overall network performance and supporting sustainable, green network implementations. Alongside resource optimization, the thesis also tackles the security implications of NF sharing. We introduce SliceScore, a DDoS filtering mechanism tailored to network slicing with NF sharing awareness, which mitigates potential security threats while maintaining high energy efficiency and ensuring the reliability of network slices.

5.1. Future Work

The contributions of this research provide a foundation for the design of next-generation networks that are not only efficient and cost-effective but also secure against emerging DDoS vulnerabilities. We encourage researchers to focus on enhancing the scalability of the proposed solutions in dynamic, real-time environments, as well as further investigating the impact of various network parameters—such as bandwidth and delay—on NF sharing. Additionally, exploring the use of machine learning to predict network behavior and optimize resource allocation can lead to even more adaptive and efficient solutions. Finally, expanding the security aspects of NF sharing, including a

broader range of attack vectors and resilience strategies, will be crucial as the networks continue to evolve.

In conclusion, this thesis intends to pave the way for next-generation communication networks that are cost-efficient, secure, and adaptable, and it opens up several promising avenues for future research in the areas of network slicing, NF sharing, and security optimization.



REFERENCES

1. Kalkan, K. and F. Alagöz, “A Distributed Filtering Mechanism Against DDoS Attacks: ScoreForCore”, *Computer Networks*, Vol. 108, pp. 199–209, 2016.
2. Jo, J. H., P. K. Sharma, J. C. S. Sicato and J. H. Park, “Emerging Technologies for Sustainable Smart City Network Security: Issues, Challenges, and Countermeasures”, *Journal of Information Processing Systems*, Vol. 15, No. 4, pp. 765–784, 2019.
3. Saberi, S., M. Kouhizadeh, J. Sarkis and L. Shen, “Blockchain Technology and Its Relationships to Sustainable Supply Chain Management”, *International Journal of Production Research*, Vol. 57, No. 7, pp. 2117–2135, 2019.
4. Seuring, S., J. Sarkis, M. Müller and P. Rao, “Sustainability and Supply Chain Management: An Introduction to the Special Issue”, *Journal of Cleaner Production*, Vol. 16, No. 15, pp. 1545–1551, 2008.
5. Yılmaz, Ö. Z., “SliceScheduler”, 2024, <https://github.com/omerzekvan/SliceScheduler>, accessed on January 7, 2025.
6. Zharabad, A. J., S. Yousefi and T. Kunz, “Network Slicing in Virtualized 5G Core with VNF Sharing”, *Journal of Network and Computer Applications*, Vol. 215, p. 103631, 2023.
7. Mohamad, A. and H. S. Hassanein, “On Demonstrating the Gain of SFC Placement with VNF Sharing at the Edge”, *2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, Hawaii, USA, 2019.
8. Series, M., “IMT Vision–Framework and Overall Objectives of the Future Development of IMT for 2020 and Beyond”, *Recommendation ITU*, Vol. 2083, pp. 1–21, 2015.

9. Benzaid, C. and T. Taleb, “AI-Driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions”, *IEEE Network*, Vol. 34, No. 2, pp. 186–194, 2020.
10. Wu, W., C. Zhou, M. Li, H. Wu, H. Zhou, N. Zhang, X. S. Shen and W. Zhuang, “AI-Native Network Slicing for 6G Networks”, *IEEE Wireless Communications*, Vol. 29, No. 1, pp. 96–103, 2022.
11. Arouk, O. and N. Nikaein, “Kube5g: A Cloud-Native 5G Service Platform”, *GLOBECOM 2020-2020 IEEE Global Communications Conference, Taipei, Taiwan*, pp. 1–6, 2020.
12. Brewer, E. A., “Kubernetes and the Path to Cloud Native”, *Proceedings of the Sixth ACM Symposium on Cloud Computing, Hawaii, USA*, pp. 167–167, 2015.
13. Carrión, C., “Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges”, *ACM Computing Surveys*, Vol. 55, No. 7, pp. 1–37, 2022.
14. Vayghan, L. A., M. A. Saied, M. Toeroe and F. Khendek, “Microservice Based Architecture: Towards High-Availability for Stateful Applications with Kubernetes”, *2019 IEEE 19th International Conference on Software Quality, Reliability and Security, Sofia, Bulgaria*, pp. 176–185, 2019.
15. Poulton, N., *The Kubernetes Book*, NIGEL POULTON LTD, 2023.
16. 3GPP, “3GPP TS 32.130 V13.0.0 (2016-01): Telecommunication Management; Network Sharing; Concepts and Requirements”, 2016, accessed on January 21, 2025.
17. Foukas, X., 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.

18. Barakabitze, A. A., A. Ahmad, R. Mijumbi and A. Hines, “5G Network Slicing Using SDN and NFV: A Survey of Taxonomy, Architectures and Future Challenges”, *Computer Networks*, Vol. 167, p. 106984, 2020.
19. Olimid, R. F. and G. Nencioni, “5G Network Slicing: A Security Overview”, *IEEE Access*, Vol. 8, pp. 99999–100009, 2020.
20. Mijumbi, R., 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, 2015.
21. Kreutz, D., F. M. Ramos, P. E. Verissimo, 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, 2014.
22. Dietrich, D., C. Papagianni, P. Papadimitriou and J. S. Baras, “Network Function Placement on Virtualized Cellular Cores”, *9th International Conference on Communication Systems and Networks (COMSNETS), Bangalore, India*, pp. 259–266, 2017.
23. Chen, X., L. Rupprecht, R. Osman, P. Pietzuch, F. Franciosi and W. Knottenbelt, “CloudScope: Diagnosing and Managing Performance Interference in Multi-Tenant Clouds”, *2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Georgia Tech Campus in Atlanta, Georgia, USA*, pp. 164–173, 2015.
24. Bushnaq, O. M., I. Zhilin, G. De Masi, E. Natalizio and I. Akyildiz, “Automatic Network Slicing for Resource Allocation in Underwater Acoustic Communication Systems”, *2021 International Balkan Conference on Communications and Networking (BalkanCom), Novi Sad, Serbia*, pp. 90–95, 2021.
25. Arora, S. and A. Ksentini, “Dynamic Resource Allocation and Placement of Cloud Native Network Services”, *ICC 2021 - IEEE International Conference on Com-*

- munications, Montréal in Montreal, Quebec, Canada*, pp. 1–6, 2021.
26. Truong-Huu, T., P. M. Mohan and M. Gurusamy, “Service Chain Embedding for Diversified 5G Slices with Virtual Network Function Sharing”, *IEEE Communications Letters*, Vol. 23, No. 5, pp. 826–829, 2019.
 27. Vittal, S. and A. A. Franklin, “HARNESS: High Availability Supportive Self Reliant Network Slicing in 5G Networks”, *IEEE Transactions on Network and Service Management*, Vol. 19, No. 3, pp. 1951–1964, 2022.
 28. Basu, D., S. Kal, U. Ghosh and R. Datta, “SoftDrone: Softwarized 5G Assisted Drone Networks for Dynamic Resource Sharing Using Machine Learning Techniques”, *Computers and Electrical Engineering*, Vol. 101, p. 107962, 2022.
 29. Yi, B., X. Wang, M. Huang, S. K. Das and K. Li, “Fairness-Aware VNF Sharing and Rate Coordination for High Efficient Service Scheduling”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 33, No. 12, pp. 4597–4611, 2022.
 30. Chergui, H. and C. Verikoukis, “Offline Service Level Agreement-Constrained Deep Learning for 5G Networks Reliable and Dynamic End-to-End Slicing”, *IEEE Journal on Selected Areas in Communications*, Vol. 38, No. 2, pp. 350–360, 2019.
 31. Liyanage, M., Q.-V. Pham, K. Dev, S. Bhattacharya, P. K. R. Maddikunta, T. R. Gadekallu and G. Yenduri, “A Survey on Zero Touch Network and Service Management for 5G and Beyond Networks”, *Journal of Network and Computer Applications*, Vol. 203, p. 103362, 2022.
 32. Gutierrez-Estevez, D. M., M. Gramaglia, A. De Domenico, G. Dandachi, S. Khatibi, D. Tsolkas, I. Balan, A. Garcia-Saavedra, U. Elzur and Y. Wang, “Artificial Intelligence for Elastic Management and Orchestration of 5G Networks”, *IEEE Wireless Communications*, Vol. 26, No. 5, pp. 134–141, 2019.

33. Gupta, A. and R. K. Jha, “A Survey of 5G Network: Architecture and Emerging Technologies”, *IEEE Access*, Vol. 3, pp. 1206–1232, 2015.
34. Frangoudis, P. A., L. Yala and A. Ksentini, “CDN-as-a-Service Provision Over a Telecom Operator’s Cloud”, *IEEE Transactions on Network and Service Management*, Vol. 14, No. 3, pp. 702–716, 2017.
35. De Alwis, C., P. Porambage, K. Dev, T. R. Gadekallu and M. Liyanage, “A Survey on Network Slicing Security: Attacks, Challenges, Solutions and Research Directions”, *IEEE Communications Surveys & Tutorials*, Vol. 26, No. 1, pp. 534–570, 2023.
36. Olimid, R. F. and G. Nencioni, “5G Network Slicing: A Security Overview”, *IEEE Access*, Vol. 8, pp. 99999–100009, 2020.
37. Van Giang, N. and Y. H. Kim, “Slicing the Next Mobile Packet Core Network”, *2014 11th International Symposium on Wireless Communications Systems, Barcelona, Spain*, pp. 901–904, 2014.
38. Yılmaz, Ö. Z. and F. Alagöz, “Shared Network Function Placement for Cloud Native 5G Core Networks”, *2023 14th International Conference on Network of the Future, Izmir, Turkey*, pp. 103–107, 2023.
39. Yılmaz, Ö. Z. and F. Alagöz, “Enabling Green and Delay-Aware 5G Edge Through Shared Cloud-Native Network Functions”, *IEEE Transactions on Green Communications and Networking*, Vol. Early Access, pp. 1–1, 2024.
40. Ghosh, S. and C. Banerjee, “Priority Based Modified Throttled Algorithm in Cloud Computing”, *2016 International Conference on Inventive Computation Technologies ,Coimbatore, Tamil Nadu, India.*, Vol. 3, pp. 1–6, 2016.
41. Yang, S., F. Li, S. Trajanovski, X. Chen, Y. Wang and X. Fu, “Delay-Aware Virtual Network Function Placement and Routing in Edge Clouds”, *IEEE Trans-*

- actions on Mobile Computing*, Vol. 20, No. 2, pp. 445–459, 2019.
42. Cacchiani, V., M. Iori, A. Locatelli and S. Martello, “Knapsack Problems—An Overview of Recent Advances. Part II: Multiple, Multidimensional, and Quadratic Knapsack Problems”, *Computers & Operations Research*, Vol. 143, p. 105693, 2022.
 43. Amaldi, E., S. Coniglio, A. M. Koster and M. Tieves, “On the Computational Complexity of the Virtual Network Embedding Problem”, *Electronic Notes in Discrete Mathematics*, Vol. 52, pp. 213–220, 2016.
 44. Vimont, Y., S. Boussier and M. Vasquez, “Reduced Costs Propagation in an Efficient Implicit Enumeration for the 01 Multidimensional Knapsack Problem”, *Journal of Combinatorial Optimization*, Vol. 15, No. 2, pp. 165–178, 2008.
 45. Panigrahy, R., K. Talwar, L. Uyeda and U. Wieder, “Heuristics for Vector Bin Packing”, 2011, <https://www.microsoft.com/en-us/research/wp-content/uploads/2011/01/VBPackingESA11.pdf>, accessed on January 21, 2025.
 46. Ang, J. S., C. Cao and H.-Q. Ye, “Model and Algorithms for Multi-Period Sea Cargo Mix Problem”, *European Journal of Operational Research*, Vol. 180, No. 3, pp. 1381–1393, 2007.
 47. 3GPP, M., “3GPP TR 38.801 V14.0.0 (2017-03): Study on New Radio Access Technology: Radio Access Architecture and Interfaces”, 2017, accessed on January 21, 2025.
 48. Nguyen, H. T., T. Van Do and C. Rotter, “Scaling User Plane Function Instances in 5G/6G Core with Deep Reinforcement Learning”, *IEEE Access*, Vol. 9, pp. 165892–165906, 2021.
 49. Alliance, N., “5G Security Recommendations Package# 2: Network Slicing”,

2016, accessed on January 21, 2025.

50. Mesbahi, M. R., M. Hashemi and A. M. Rahmani, “Performance Evaluation and Analysis of Load Balancing Algorithms in Cloud Computing Environments”, *2016 Second International Conference on Web Research, Tehran, Iran*, pp. 145–151, 2016.
51. Afolabi, I., T. Taleb, K. Samdanis, A. Ksentini and H. Flinck, “Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions”, *IEEE Communications Surveys & Tutorials*, Vol. 20, No. 3, pp. 2429–2453, 2018.
52. Kuadey, N. A. E., G. T. Maale, T. Kwantwi, G. Sun and G. Liu, “DeepSecure: Detection of Distributed Denial of Service Attacks on 5G Network Slicing—Deep Learning Approach”, *IEEE Wireless Communications Letters*, Vol. 11, No. 3, pp. 488–492, 2021.
53. Schinianakis, D., R. Trapero, D. S. Michalopoulos and B. G.-N. Crespo, “Security Considerations in 5G Networks: A Slice-Aware Trust Zone Approach”, *2019 IEEE Wireless Communications and Networking Conference, Marrakesh, Morocco*, pp. 1–8, 2019.
54. Javadpour, A., F. Ja’fari, T. Taleb and C. Benzaïd, “Reinforcement Learning-Based Slice Isolation Against Distributed Denial of Service Attacks in Beyond 5G Networks”, *IEEE Transactions on Network and Service Management*, Vol. 20, No. 3, pp. 3930–3946, 2023.
55. Sattar, D. and A. Matrawy, “Towards Secure Slicing: Using Slice Isolation to Mitigate DDoS Attacks on 5G Core Network Slices”, *2019 IEEE Conference on Communications and Network Security, Washington, DC, USA*, pp. 82–90, 2019.
56. Hewa, T., A. Kalla, P. Porambage, M. Liyanage and M. Ylianttila, “How DoS Attacks Can Be Mounted on Network Slice Broker and Can They Be Mitigated

- Using Blockchain?”, *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications, Helsinki, Finland*, pp. 1525–1531, 2021.
57. Thantharate, A., R. Paropkari, V. Walunj, C. Beard and P. Kankariya, “Secure5G: A Deep Learning Framework Towards a Secure Network Slicing in 5G and Beyond”, *2020 10th Annual Computing and Communication Workshop and Conference, Las Vegas, Nevada, USA*, pp. 0852–0857, 2020.
 58. Li, Y., Y. Zhao, J. Li, J. Zhang, X. Yu and J. Zhang, “Side Channel Attack-Aware Resource Allocation for URLLC and eMBB Slices in 5G RAN”, *IEEE Access*, Vol. 8, pp. 2090–2099, 2019.
 59. Kotulski, Z., T. Nowak, M. Sepczuk, M. Tunia, R. Artych, K. Bocianiak, T. Osko and J.-P. Wary, “On End-to-End Approach for Slice Isolation in 5G Networks. Fundamental Challenges”, *2017 Federated Conference on Computer Science and Information Systems, Prague, Czech Republic*, pp. 783–792, 2017.
 60. Chen, B., S. Qiao, J. Zhao, D. Liu, X. Shi, M. Lyu, H. Chen, H. Lu and Y. Zhai, “A Security Awareness and Protection System for 5G Smart Healthcare Based on Zero-Trust Architecture”, *IEEE Internet of Things journal*, Vol. 8, No. 13, pp. 10248–10263, 2021.
 61. Wang, G., G. Feng, T. Q. Quek, S. Qin, R. Wen and W. Tan, “Reconfiguration in Network Slicing—Optimizing the Profit and Performance”, *IEEE Transactions on Network and Service Management*, Vol. 16, No. 2, pp. 591–605, 2019.
 62. Marquez, C., M. Gramaglia, M. Fiore, A. Banchs and X. Costa-Perez, “Resource Sharing Efficiency in Network Slicing”, *IEEE Transactions on Network and Service Management*, Vol. 16, No. 3, pp. 909–923, 2019.
 63. Kocher, P., “Differential Power Analysis”, *Proc. Advances in Cryptology (CRYPTO’99), Santa Barbara, California, USA*, 1999.

64. Devi, M. and A. Majumder, “Side-Channel Attack in Internet of Things: A Survey”, *Applications of Internet of Things: Proceedings of ICCCIOT 2020, Agartala, Tripura, India*, pp. 213–222, 2021.
65. Cáceres-Hidalgo, J. and D. Avila-Pesantez, “Cybersecurity Study in 5G Network Slicing Technology: A Systematic Mapping Review”, *2021 IEEE Fifth Ecuador Technical Chapters Meeting, Cuenca, Ecuador*, pp. 1–6, 2021.
66. Zhang, Y., A. Juels, M. K. Reiter and T. Ristenpart, “Cross-VM Side Channels and Their Use to Extract Private Keys”, *Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, North Carolina, USA*, pp. 305–316, 2012.
67. Yarom, Y. and K. Falkner, “{FLUSH+ RELOAD}: A High Resolution, Low Noise, L3 Cache {Side-Channel} Attack”, *23rd USENIX Security Symposium (USENIX Security 14), Anaheim, California, USA*, pp. 719–732, 2014.
68. Ge, Q., Y. Yarom, D. Cock and G. Heiser, “A Survey of Microarchitectural Timing Attacks and Countermeasures on Contemporary Hardware”, *Journal of Cryptographic Engineering*, Vol. 8, pp. 1–27, 2018.
69. Liu, F., Q. Ge, Y. Yarom, F. Mckeen, C. Rozas, G. Heiser and R. B. Lee, “Catalyst: Defeating Last-Level Cache Side Channel Attacks in Cloud Computing”, *2016 IEEE International Symposium on High Performance Computer Architecture, Barcelona, Spain*, pp. 406–418, 2016.
70. Costan, V., I. Lebedev and S. Devadas, “Sanctum: Minimal Hardware Extensions for Strong Software Isolation”, *25th USENIX Security Symposium (USENIX Security 16), Seattle, Washington, USA*, pp. 857–874, 2016.
71. Yilmaz, O. Z., K. Kalkan and F. Alag”oz, “SliceScore: A Network Function Sharing Aware and Slice-Oriented DDoS Filtering Approach”, *IEEE Access*, Vol. 13, pp. 18294–18307, 2025.

72. Kholidy, H. A., A. Karam, J. Sidoran, M. A. Rahman, M. Mahmoud, M. Badr, M. Mahmud and A. F. Sayed, "Toward Zero Trust Security in 5G Open Architecture Network Slices", *MILCOM 2022 - 2022 IEEE Military Communications Conference, Rockville, Maryland, USA*, pp. 577–582, 2022.
73. Vittal, S., U. Dixit, S. P. Sovitkar, K. Sowjanya and A. A. Franklin, "Preventing Cross Network Slice Disruptions in a Zero-Trust and Multi-Tenant Future 5G Networks", *2023 IEEE 9th International Conference on Network Softwarization, Madrid, Spain*, pp. 227–231, 2023.
74. Chen, C.-H., C. Chen, S.-H. Lu and C.-C. Tseng, "Role-Based Campus Network Slicing", *2016 IEEE 24th International Conference on Network Protocols, Singapore*, pp. 1–6, 2016.
75. Zhang, R., M. Song, T. Li, Z. Yu, Y. Dai, X. Liu and G. Wang, "Democratic Learning: Hardware/Software Co-design for Lightweight Blockchain-Secured On-Device Machine Learning", *Journal of Systems Architecture*, Vol. 118, p. 102205, 2021.
76. Kalkan, K., L. Altay, G. Gür and F. Alagöz, "JESS: Joint Entropy-Based DDoS Defense Scheme in SDN", *IEEE Journal on Selected Areas in Communications*, Vol. 36, No. 10, pp. 2358–2372, 2018.
77. Kim, Y., W. C. Lau, M. C. Chuah and H. J. Chao, "PacketScore: A Statistics-Based Packet Filtering Scheme Against Distributed Denial-of-Service Attacks", *IEEE Transactions on Dependable and Secure Computing*, Vol. 3, No. 2, pp. 141–155, 2006.
78. Jin, C., H. Wang and K. G. Shin, "Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic", *Proceedings of the 10th ACM Conference on Computer and Communications Security, Washington, DC, USA*, pp. 30–41, 2003.

79. Udhayan, J. and T. Hamsapriya, “Statistical Segregation Method to Minimize the False Detections During DDoS Attacks”, *International Journal of Network Security*, Vol. 13, No. 3, pp. 152–160, 2011.
80. Coronado, E., R. Riggio, J. Villa1ón and A. Garrido, “Lasagna: Programming Abstractions for End-to-End Slicing in Software-Defined WLANs”, *2018 IEEE 19th International Symposium on “A World of Wireless, Mobile and Multimedia Networks”*, Chania, Greece, pp. 14–15, 2018.
81. 3GPP, M., “3GPP TR 38.413 V15.13.0 (2021-10): Next-Generation Radio Access Networks; Next-Generation Application Protocol”, 2021, accessed on January 21, 2025.
82. Zhao, Y., P. Li and C. Hu, “Verification and Evaluation on 5G Multi-Operator Core Network”, *2021 International Wireless Communications and Mobile Computing*, Harbin, China, pp. 498–502, 2021.
83. Cao, K., B. Wang, H. Ding, L. Lv, R. Dong, T. Cheng and F. Gong, “Improving Physical Layer Security of Uplink NOMA via Energy Harvesting Jammers”, *IEEE Transactions on Information Forensics and Security*, Vol. 16, pp. 786–799, 2021.
84. Kalkan, K., G. Gur and F. Alagoz, “Defense Mechanisms Against Distributed Denial of Service Attacks in Software Defined Networking Environment”, *IEEE Communications Magazine*, Vol. 55, No. 9, pp. 175–179, 2017.
85. Suh, K., S. Kim, Y. Ahn, S. Kim, H. Ju and B. Shim, “Deep Reinforcement Learning-Based Network Slicing for Beyond 5G”, *IEEE Access*, Vol. 10, pp. 7384–7395, 2022.
86. Su, R., D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang and Z. Zhu, “Resource Allocation for Network Slicing in 5G Telecommunication Networks: A Survey of Principles and Models”, *IEEE Network*, Vol. 33, No. 6, pp. 172–179, 2019.

87. Kaloxylos, A., “A Survey and an Analysis of Network Slicing in 5G Networks”, *IEEE Communications Standards Magazine*, Vol. 2, No. 1, pp. 60–65, 2018.
88. Cunha, J., P. Ferreira, E. M. Castro, P. C. Oliveira, M. J. Nicolau, I. Núñez, X. R. Sousa and C. Serôdio, “Enhancing Network Slicing Security: Machine Learning, Software-Defined Networking, and Network Functions Virtualization-Driven Strategies”, *Future Internet*, Vol. 16, No. 7, p. 226, 2024.
89. Li, G., H. Chen, L. Wu, X. Chi, J. Yao, F. Xia and J. Yu, “Efficient Deployment and Scheduling of Shared VNF Instances in Mobile Edge Computing Networks”, *IEEE Internet of Thingsjournal*, Vol. 11, No. 19, pp. 32259 – 32271, 2024.
90. Martins, J. S., T. C. Carvalho, R. Moreira, C. B. Both, A. Donatti, J. H. Corrêa, J. A. Suruagy, S. L. Corrêa, A. J. Abelem, M. R. Ribeiro *et al.*, “Enhancing Network Slicing Architectures with Machine Learning, Security, Sustainability and Experimental Networks Integration”, *IEEE Access*, Vol. 11, pp. 69144–69163, 2023.
91. Wright, P., C. White, R. C. Parker, J.-S. Pegon, M. Menchetti, J. Pearce, A. Bahrami, A. Moroz, A. Wonfor, R. V. Pentty *et al.*, “5G Network Slicing with QKD and Quantum-Safe Security”, *Journal of Optical Communications and Networking*, Vol. 13, No. 3, pp. 33–40, 2021.
92. Salahdine, F., Q. Liu and T. Han, “Towards Secure and Intelligent Network Slicing for 5G Networks”, *IEEE Openjournal of the Computer Society*, Vol. 3, pp. 23–38, 2022.
93. Wichary, T., J. Mongay Batalla, C. X. Mavromoustakis, J. Żurek and G. Mastroakis, “Network Slicing Security Controls and Assurance for Verticals”, *Electronics*, Vol. 11, No. 2, p. 222, 2022.
94. Dangi, R., A. Jadhav, G. Choudhary, N. Dragoni, M. K. Mishra and P. Lalwani, “ML-Based 5G Network Slicing Security: A Comprehensive Survey”, *Future In-*

- ternet*, Vol. 14, No. 4, p. 116, 2022.
95. Firoozjaei, M. D., J. P. Jeong, H. Ko and H. Kim, “Security Challenges with Network Functions Virtualization”, *Future Generation Computer Systems*, Vol. 67, pp. 315–324, 2017.
 96. Zhang, R., X. Cheng and L. Yang, “Cooperation via Spectrum Sharing for Physical Layer Security in Device-to-Device Communications Underlying Cellular Networks”, *IEEE Transactions on Wireless Communications*, Vol. 15, No. 8, pp. 5651–5663, 2016.
 97. Park, J.-M., J. H. Reed, A. Beex, T. C. Clancy, V. Kumar and B. Bahrak, “Security and Enforcement in Spectrum Sharing”, *Proceedings of the IEEE*, Vol. 102, No. 3, pp. 270–281, 2014.
 98. Pattaranantakul, M., R. He, A. Meddahi and Z. Zhang, “SecMANO: Towards Network Functions Virtualization (NFV) Based Security Management and Orchestration”, *2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China*, pp. 598–605, 2016.
 99. Kasera, S., J. Pinheiro, C. Loader, M. Karaul, A. Hari and T. LaPorta, “Fast and Robust Signaling Overload Control”, *Proceedings Ninth International Conference on Network Protocols. ICNP 2001, Riverside, California, USA*, pp. 323–331, 2001.
 100. Bisht, H., M. Patra and S. Kumar, “Detection and Localization of DDoS Attack During Inter-Slice Handover in 5G Network Slicing”, *2023 IEEE 20th Consumer Communications & Networking Conference, Las Vegas, Nevada, USA*, pp. 798–803, 2023.
 101. Khan, M. S., B. Farzaneh, N. Shahriar, N. Saha and R. Boutaba, “SliceSecure: Impact and Detection of Distributed Denial of Service Attacks on 5G Network Slices”, *2022 IEEE Future Networks World Forum, Montreal, Canada*, pp. 639–642, 2022.

102. Boualouache, A., T. E. T. Djaidja, S.-M. Senouci, Y. Ghamri-Doudane, B. Brik and T. Engel, “Deep Learning-Based Intra-Slice Attack Detection for 5G-V2X Sliced Networks”, *2022 IEEE 95th Vehicular Technology Conference (VTC2022-Spring)*, Tehran, Iran, pp. 1–5, 2022.
103. Sathi, V. N. and C. S. R. Murthy, “Distributed Slice Mobility Attack: A Novel Targeted Attack Against Network Slices of 5G Networks”, *IEEE Networking Letters*, Vol. 3, No. 1, pp. 5–9, 2020.
104. Grant, S., A. Yelam, M. Bland and A. C. Snoeren, “SmartNIC Performance Isolation with FairNIC: Programmable Networking for the Cloud”, *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, New York, USA, pp. 681–693, 2020.
105. Yigit, B., G. Gur, B. Tellenbach and F. Alagoz, “Secured Communication Channels in Software-Defined Networks”, *IEEE Communications Magazine*, Vol. 57, No. 10, pp. 63–69, 2019.
106. Li, H., Z. Kong, Y. Chen, L. Wang, Z. Lu, X. Wen, W. Jing and W. Xiang, “Slice-Based Service Function Chain Embedding for End-to-End Network Slice Deployment”, *IEEE Transactions on Network and Service Management*, Vol. 20, No. 3, pp. 3652 – 3672, 2023.
107. Duytam Ly, L., M. Sadeghi Ghahroudi and V. Ponce, “A Systematic Literature Review of Reliable Provisioning for Virtual Network Function Chaining”, *Applied Sciences*, Vol. 13, No. 9, p. 5504, 2023.
108. Al-Saadi, R., G. Armitage, J. But and P. Branch, “A Survey of Delay-Based and Hybrid TCP Congestion Control Algorithms”, *IEEE Communications Surveys & Tutorials*, Vol. 21, No. 4, pp. 3609–3638, 2019.
109. Kellerer, H., U. Pferschy, D. Pisinger, H. Kellerer, U. Pferschy and D. Pisinger,

Multidimensional Knapsack Problems, Springer, 2004.

110. Han, B., V. Gopalakrishnan, L. Ji and S. Lee, “Network Function Virtualization: Challenges and Opportunities for Innovations”, *IEEE Communications Magazine*, Vol. 53, No. 2, pp. 90–97, 2015.
111. Mehraghdam, S., M. Keller and H. Karl, “Specifying and Placing Chains of Virtual Network Functions”, *2014 IEEE 3rd International Conference on Cloud Networking, Luxembourg City, Luxembourg*, pp. 7–13, 2014.
112. Sajjad, M. M., C. J. Bernardos, D. Jayalath and Y.-C. Tian, “Inter-Slice Mobility Management in 5G: Motivations, Standard Principles, Challenges, and Research Directions”, *IEEE Communications Standards Magazine*, Vol. 6, No. 1, pp. 93–100, 2022.
113. Dräxler, S., M. Keller and H. Karl, “Specifying and Placing Chains of Virtual Network Functions”, *Proceedings of the 3rd International Conference on Cloud Networking, Luxembourg*, pp. 7–13, 2014.
114. Zhu, D., “Max–Min Bin Packing Algorithm and Its Application in Nano-Particles Filling”, *Chaos, Solitons & Fractals*, Vol. 89, pp. 83–90, 2016.
115. Pino, A., P. Khodashenas, X. Hesselbach, E. Coronado and S. Siddiqui, “Validation and Benchmarking of CNFs in OSM for Pure Cloud Native Applications in 5G and Beyond”, *2021 International Conference on Computer Communications and Networks, Athens, Greece*, pp. 1–9, 2021.
116. Open, I. S. E., “5G Private Wireless Experience Kit with Integrated Radio Access Networks”, 2023, <https://smart-edge-open.github.io/ido-specs/doc/reference-architectures/smartedge-open-experience-kit-pwek-all-in-one>, accessed on January 14, 2024.
117. Isotalo, L., “5G Slicing as a Tool to Test User Equipment Against Advanced

- Persistent Threats”, *International Conference on Network and System Security*, Abu Dhabi, United Arab Emirates, pp. 595–603, 2017.
118. Samdanis, K., X. Costa-Perez and V. Sciancalepore, “From Network Sharing to Multi-Tenancy: The 5G Network Slice Broker”, *IEEE Communications Magazine*, Vol. 54, No. 7, pp. 32–39, 2016.
119. Soenen, T., R. Banerjee, W. Tavernier, D. Colle and M. Pickavet, “Demystifying Network Slicing: From Theory to Practice”, *IFIP/IEEE Symposium on Integrated Network and Service Management*, Lisbon, Portugal, pp. 1115–1120, 2017.