DOCTORAL THESIS

Radio Access Network Subslicing in Closed Control Loop for Improved Slice Performance Management

Marika Kulmar

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
TALLINN 2024

TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
2/2024

Radio Access Network Subslicing in Closed Control Loop for Improved Slice Performance Management

MARIKA KULMAR

TAL
TECH
PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Thomas Johann Seebeck Department of Electronics

The dissertation was accepted for the defence of the degree of Doctor of Philosophy (Information and Communication Technology) on 18 December 2023

Supervisor: Professor Muhammad Mahtab Alam,
Thomas Johann Seebeck Department of Electronics
School of Information Technology
Tallinn University of Technology
Tallinn, Estonia

Co-supervisor: Dr. Ivo Müürsepp,
Thomas Johann Seebeck Department of Electronics
School of Information Technology
Tallinn University of Technology
Tallinn, Estonia

Co-supervisor: Dr. Sven Pärand,
Telia Estonia Ltd
Tallinn, Estonia

Opponents: Professor Olivier Berder,
University of Rennes, CNRS, IRISA
Lannion, France

Professor Luca Reggiani,
Politecnico Di Milano, Department of Electronics, Information and Bioengineering
Milano, Italy

Defence of the thesis: 26 January 2024, Tallinn

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology, has not been submitted for any academic degree elsewhere.

Marika Kulmar

signature

Copyright: Marika Kulmar, 2024

ISSN 2585-6898 (publication)
ISBN 978-9916-80-096-6 (publication)
ISSN 2585-6901 (PDF)
ISBN 978-9916-80-097-3 (PDF)
DOI https://doi.org/10.23658/taltech.2/2024
Printed by Koopia Niini & Rauam

Kulmar, M. (2024). *Radio Access Network Subslicing in Closed Control Loop for Improved Slice Performance Management* [TalTech Press]. https://doi.org/10.23658/taltech.2/2024

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
2/2024

Raadiojuurdepääsu võrguviilu alamviilutamine suletud juhtimisahelas

MARIKA KULMAR

TAL
TECH
KIRJASTUS

Contents

List of Publications 8

Author's Contributions to the Publications 9

Abbreviations 10

1 Introduction 13
- 1.1 Motivation 15
- 1.2 Problem Statement 16
- 1.3 Research Questions 16
- 1.4 Contributions 16
- 1.5 Thesis Organization 17

2 Background and State of the Art 19
- 2.1 Network Slicing 19
  - 2.1.1 Core Slicing 19
  - 2.1.2 RAN Slicing 19
  - 2.1.3 Network Slice Template 21
  - 2.1.4 Network Slice Identifier 21
  - 2.1.5 Management of Slices 22
  - 2.1.6 Management of Resources 24
  - 2.1.7 Automation in Management 25
  - 2.1.8 Summary 26
- 2.2 RAN Schedulers (Radio Resource Allocation) 26
  - 2.2.1 Schedulers comparison 28
  - 2.2.2 Summary 31
- 2.3 Slice Life Cycle (Slice Provision) 31
  - 2.3.1 Slice preparation and creation (provision) 31
  - 2.3.2 Slice modification (performance assurance) 33
  - 2.3.3 Slice deletion 34
  - 2.3.4 Summary 35
- 2.4 Subslicing 35
  - 2.4.1 Summary 36
- 2.5 5G RAN NR Emulators and Simulators 36
  - 2.5.1 Emulators 36
  - 2.5.2 Simulators 38
  - 2.5.3 Summary 41
- 2.6 Conclusion 41

3 Evaluation of the RAN Slice Performance 42
- 3.1 Simulation Planning 42
  - 3.1.1 Simulation Tool 42
  - 3.1.2 Simulation Time 42
  - 3.1.3 Subslice Sizes 42
  - 3.1.4 Carrier Frequency 43
  - 3.1.5 UE Requested Rates 43
  - 3.1.6 UE Packet Sizes 43
  - 3.1.7 UE BLER 43

    - 3.1.8 Other Settings 43
    - 3.1.9 KPIs to be Collected 44
  - 3.2 Results 44
    - 3.2.1 Dataset 1 45
    - 3.2.2 Dataset 2 46
    - 3.2.3 Findings 46
  - 3.3 Conclusion 48

4 RAN Subslicing Algorithms 49
  - 4.1 Minimum Subslice Size Requirement 49
  - 4.2 UE Clustering 49
  - 4.3 Bandwidth Allocation 50
  - 4.4 Evaluation of the Subslicing Algorithms 51
    - 4.4.1 Effect of the Minimum Subslice Size 52
    - 4.4.2 Effect of Subslicing for Slice Containing Set of UEs With Different BLERs 53
    - 4.4.3 Effect of UE Clustering by Rate and BLER 54
  - 4.5 Conclusion 54

5 Enhanced Decision Mechanisms for RAN Subslicing 56
  - 5.1 Training Data Labeling 56
    - 5.1.1 Performance Data Clustering 56
    - 5.1.2 Discussion of Deciding Subslice Operation by Clustering Result 56
    - 5.1.3 Optimization Problem to Find Best Subslice Size 57
  - 5.2 Neural Network and Search of Hyperparameters 59
  - 5.3 Conclusion 59

6 Subslicing in Closed Control Loop to Improve Slice Performance 60
  - 6.1 Management Closed Control Loop Architecture 60
    - 6.1.1 Monitor 60
    - 6.1.2 Analyze 61
    - 6.1.3 Decide (Plan) 61
    - 6.1.4 Execute 61
  - 6.2 Evaluation 61
    - 6.2.1 Simulation Setup 61
    - 6.2.2 Scenarios 61
    - 6.2.3 Results of Initialization 62
    - 6.2.4 Results of Runtime, Scenario 1 63
    - 6.2.5 Results of Runtime, Scenario 2 63
    - 6.2.6 Discussion of Possible Improvements 66
  - 6.3 Conclusion 67

7 Conclusion and Future Works 68
  - 7.1 Summary 68
  - 7.2 Research Questions Answered 68
  - 7.3 Contributions and Limitations 69
  - 7.4 Future Work 70

List of Figures 71

List of Tables 73
References 74
Acknowledgements 83
Abstract 84
Kokkuvõte 85
Appendix 1 87
Appendix 2 97
Appendix 3 125
Appendix 4 135
Curriculum Vitae 155
Elulookirjeldus 156

List of Publications

The present Ph.D. thesis is based on the following publications

- I M. Kulmar, I. Muursepp, and M. M. Alam, "The Impact of RAN Slice Bandwidth Subpartitioning on Slice Performance," in *2022 Int. Wirel. Commun. Mob. Comput.*, IEEE, May 2022, pp. 419–424, ISBN: 978-1-6654-6749-0, DOI: `10.1109/IWCMC55113.2022.9825262`
- II M. Kulmar, I. Müürsepp, and M. M. Alam, "Heuristic Radio Access Network Subslicing with User Clustering and Bandwidth Subpartitioning," *Sensors*, vol. 23, no. 10, p. 4613, May 2023, ISSN: 1424-8220, DOI: `10.3390/s23104613`
- III M. Kulmar, I. Müürsepp, and M. M. Alam, "Enhanced Decision Mechanism for RAN Subslicing in Management Closed Control Loop," pp. 175–181, Sep. 2023, DOI: `10.1109/FMEC59375.2023.10306223`
- IV M. Kulmar, M. M. Alam, and I. Müürsepp, "Management Closed Control Loop Automation for Improved RAN Slice Performance by Subslicing," *TechRxiv. December 18*, 2023, Submitted to IEEE Open J. Commun. Soc. (under review), DOI: `10.22541/techrxiv.170290948.88447519/v1`

Author’s Contributions to the Publications

I In my publication I, I was the main author; I wrote the simulation program, which calls the MATLAB 5G-NR simulator with my specified settings for RAN subslices. Then I carried out the simulations and the analysis of the results, prepared the figures, and wrote the manuscript.

II In my publication II, I was the main author; I proposed the UE clustering algorithm which is based on modified k-means, and the subslice bandwidth allocation algorithm. I implemented the algorithms in MATLAB to create the settings for RAN subslices. I modified my simulation program, which calls the MATLAB 5G-NR simulator with my specified settings for RAN subslices. Then I carried out the simulations and the analysis of the results, prepared the figures, and wrote the manuscript.

III In my publication III, I was the main author; I run the subslice simulations to create input data for my proposed decision mechanism. I analyzed the clusterability of the subslice performance data using MATLAB. I clustered the subslice performance data into three clusters to match the decisions of "no change", "merge" and "split" for subslices to be labelled as training data. I trained the neural network and evaluated its best settings. Then I prepared the figures, and wrote the manuscript.

IV In my publication IV, I was the main author; I implemented the Monitor-Analyze-Plan-Execute-Knowledge (MAPE-K) management closed control loop (MCCL) in MATLAB. For Plan function, I proposed to solve an optimization problem for training data labelling. For execute function, I proposed the subslice processing and the subslice splitting algorithms. I trained the decision neural network. I wrote the discrete event simulator, which calls the MATLAB 5G-NR simulator with my specified settings for RAN subslices. Then I carried out the simulations and the analysis of the results, prepared the figures, and wrote the manuscript.

Abbreviations

3GPP	3rd Generation Partnership Project, a standard definition organization
5G	Fifth Generation
5GS	5G system
5QI	5G QoS Identifier
AMF	Access and Mobility Function
AR	Augmented Reality
BBU	Base Band Unit
BDA	Big Data Analytics
BE	Best effort
BLER	Block Error Ratio
BS	Base Station
BWP	Bandwidth Part
C-RAN	Cloud/Centralized RAN
CCL	Closed Control Loop
CBR	Constant Bit Rate
CN	Core Network
COTS	Commercially available off-the-shelf
CPU	Central Processing Unit, a computing resource
CQI	Channel Quality Indicator
CSI-RS	Channel State Information Reference Signals
CSMF	Communication Service Management Function
CU	Central Unit, a constituent of a gNB
CUPS	control and user plane separation
DAS	Direct Attached Storage, a storage hardware
DB	Database
DC	Data Center
DL	downlink, from RAN to UE
DM-RS	Demodulation Reference Signals
DN	Data Network
DSbB	Dynamic Subslicing by BLER, my proposed subslice splitting algorithm
DSbR	Dynamic Subslicing by Rate, my used subslice splitting algorithm
DU	Distributed Unit, a constituent of a gNB
E2E	End to end
eMBB	enhanced Mobile Broadband
EN-DC	E-UTRA-NR Dual Connectivity
ETSI	European Telecommunications Standards Institute, a standard definition organization
FAPI	Functional Application Platform Interface, interface between gNB units
FCAPS	Fault, Configuration, Accounting, Performance, Security Management
FDD	Frequency Division Duplex
FR	Frequency Range
gNB	g Node B, a 5th generation RAN base station
GSMA	Global System for Mobile Communications (GSM) Association
GST	Generic network Slice Template
HMTC	High-Performance Machine-type Communications
IoT	Internet of Things
IQ	In-phase and Quadrature, data for signal representation

KNN	K-nearest neighbour, classification algorithm
KPI	Key Performance Indicator
LCM	Lifecycle Management
M2M	Machine to machine
MANO	Management ANd Orchestration
MAPE-K	Monitor-Analyze-Plan-Execute-Knowledge, a CCL
MBR	Minimum Bit Rate
MCCL	Management CCL
MCS	Modulation and Coding Scheme
MIoT	Massive Internet of Things
ML	Machine Learning
mMTC	massive Machine Type Communication
MVNO	Mobile Virtual Network Operator
NAS	Network Attached Storage, a storage hardware
NF	Network Function
NFV	Network Function Virtualization
NFVI	NFV Infrastructure
NN	Neural Network
NoSS	No subslicing, my abbreviation of no subslicing
NR	New Radio, evolution from LTE (4G RAN) to 5G
NSA	Non-Standalone
NSI	Network Slice Instance
NSMF	Network Slice Management Function
NSSF	Network Slice Selection Function
NSSI	Network Slice Subnet Instance
MSSMF	Network Slice Subnet Management Function
NSSP	Network Slice Selection Policy
NWDAF	NetWork Data Analytics Function
O-RAN	Open Radio Access Network, a standard definition organization
OAI	OpenAirInterface
OFDM	Orthogonal Frequency Myltiplexing
OODA	Observe-Orient-Decide-Act, a CCL
OS	Operating System
PDCP	Packet Data Convergence Protocol
PLMN	Public Land Mobile Network
PNF	Physical Network Function
PoP	Point of Presence
PRB	Physical Resource Block
QoS	Quality of Service
RAN	Radio Access Network
RB	Resource Block
ReLU	Rectified Linear Unit, activation function of NN
RF	Radio Frequency
RIC	RAN Intelligent Controller
RLC	Radio Link Control
RMSE	Root Mean Square Error
RQ	Research Question
RRC	Radio Resource Control
RRH	Remote Radio Head

RRM	Radio Resource Management
RU	Radio Unit, a constituent of a gNB
S3S	Static 3 subslices
S-NSSAI	Single Network Slice Selection Assistance Information
SAN	Storage Area Networked, a storage hardware
SCS	Subcarrier Spacing
SD	Slice Descriptor, optional part of slice identifier
SD-RAN	Software Defined RAN
SDN	Software Defined Networking
SDR	Software Defined Radio
SINR	Signal-to-interference-plus-noise ratio
SLA	Service Level Agreement
SLAW	self-similar least-action human walk, a model of UE mobility
SMF	Session Management Function
SNR	Signal-to-Noise Ratio
SST	Slice/Service Type, mandatory part of slice identifier
SVM	Support Vector Machine
TBS	Transport Block Size
TN	Transport Network, provides connectivity between CN, RAN and other domains
UCwBA	User Clustering with Bandwidth Allocation, my proposed subslicing algorithm
UE	User Equipment, mobile user device
UL	uplink, from UE to RAN
UPF	User Plane Function, a gateway
URLLC	Ultra Reliable Low Latency Communication
USRP	Universal Software Radio Peripheral, an SDR device
V2X	Vehicle to everything
VNF	Virtual Network Function
VR	Virtual Reality
WiSE	Wireless Simulator Evolution

1 Introduction

The mobile network data traffic has grown by 40% reaching to 118 Exabytes (EB) per month by the end of 2022 and is predicted to reach to 472 EB per month by the end of 2028 [5]. Notably, different applications impose varying demands on network traffic. Approximately 10% of users, largely utilizing multimedia services, account for a significant 70% of network traffic.

Low-complexity, cost-effective devices with moderate throughput requirement are connected to the cellular network in increased numbers, having reached to 500 million at the end of 2022 [5]. Internet of Things (IoT) connections to 5G networks are on the rise, driven by the phasing out of older 2G and 3G technologies. The broadband and critical IoT (4G/5G) are predicted to be 60% of all cellular IoT connections by the end of 2028 [5].

The 5th generation (5G) mobile network is meant to satisfy different requirements of new applications and services by using a single infrastructure. Each service has a service level agreement (SLA), which describes quality of service (QoS) requirements.

The services and use-cases are divided into three verticals [6]:

- The enhanced mobile broadband (eMBB) vertical collects all services, applications and use-cases that require high data rate (in Gbps) such as multimedia on big screens, smart offices, videostreams for virtual reality (VR) or augmented reality (AR) etc.

- The massive machine type communications (mMTC) vertical contains Internet of Things (IoT) network. The "things" have a massive count (million devices per km^2) of user equipments (UE) of varying complexity, energy consumption, cost, transmission power and latency requirements [7]. The smartphones, cameras, sensors, actuators etc. can be used in smart grid, agriculture, healthcare and other communication use-cases.

- The ultra-reliable low latency communication (URLLC) vertical has strict requirements of latency (≤ 1 ms) and reliability [7]. The URLLC human-centric applications are for example in health, safety and entertainment, and machine-centric URLLC application examples include industry automation, self-driving cars, emergency and disaster response, e-health and smart grids.

The single network cannot satisfy all these very different requirements, all with the same infrastructure and at the same time. In 5G, network slicing is introduced to build multiple logical networks, called slices, on top of physical infrastructure. Each slice contains a specific set of the aforementioned requirements to serve UEs, which need a specific subset of requirements to be met in a service vertical. The network slicing enables to establish mobile virtual network operators (MVNO), which do not need to build their own separate physical network infrastructure. Instead, an MVNO can use a slice of resources obtained from the infrastructure provider as an MVNO slice.

The top use cases to benefit from network slicing in 5G come from the industry segments of healthcare, government, transportation, energy & utilities, manufacturing and media & entertainment. These six industries make up 90% of the market revenues [8]. The first commercial network slicing in 5G was launched in 2022 by Singtel in Singapore. Lessons learned that the network slicing in 5G, if deployed properly, has a strong business potential for service providers and supports new use cases in limited geographical areas for customers [5].

The general architecture of 5G system (5GS) is shown in Fig. 1. 5G core network (CN) has a service-based architecture. For each network service there is a network function (NF), e.g. access and mobility management function (AMF), session management function (SMF), etc. UE connects to radio access network (RAN) and via user plane function (UPF) it has access to data network (DN) outside 5GS, the internet. For network slicing these functions are virtualized, that is, abstracted. Resources are computing, storage, networking. The goal is the resource usage efficiency with all services' requirements satisfied.

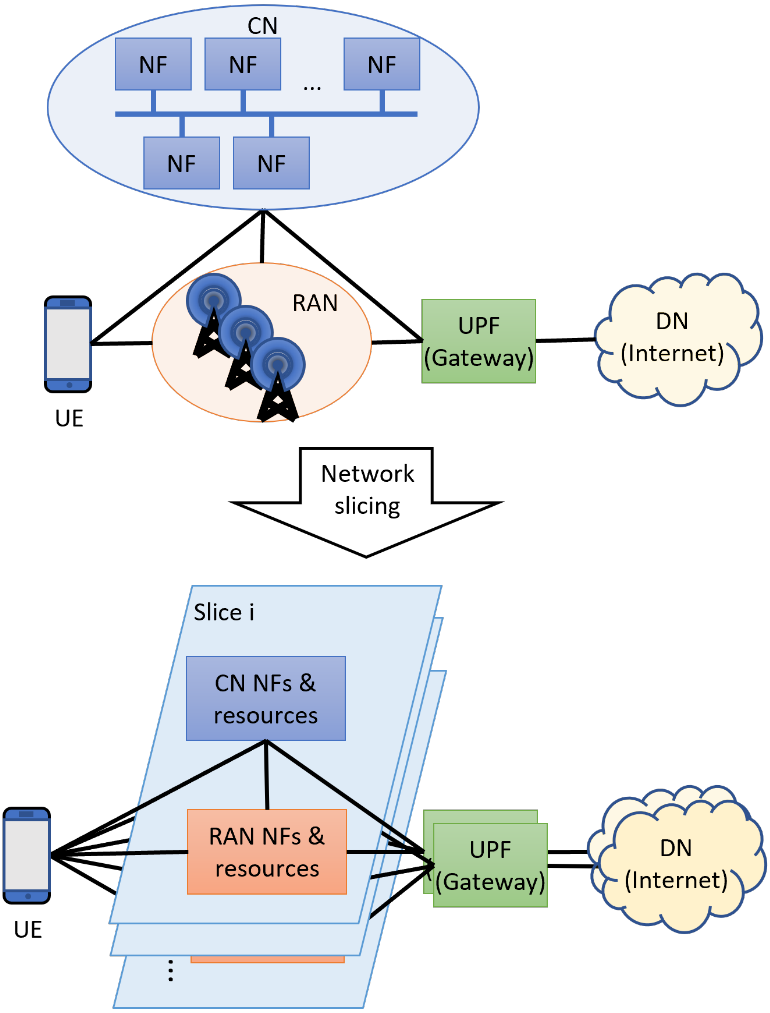


Figure 1: Architecture of 5GS

To satisfy the rate requirement of eMBB vertical, sufficient networking resources are allocated. In a radio access network (RAN), bandwidth parts (BWP) using higher frequencies in frequency range 2 (FR2, 24.25-52.6 GHz [9]) are more suitable because wide bands with subcarrier spacings of 120 kHz are available. In addition, larger subcarrier spacing enable to use wide bandwidths. The vertical mMTC requires massive connectivity, while the rate and reliability requirements are low. The low frequencies, small BWPs and subcarrier spacing are used for UEs in this vertical. The resources allocated must guarantee the requested rates and satisfy low latency. This is done by prioritizing resource allocation to this slice type.

In RAN, the base station (BS) called gNB contains software and specific hardware. The software, which does not require specific hardware, can be implemented as VNFs, similar to core virtualization. For virtualization of the gNB, its control plane and user plane are separated, and the gNB can be functionally split into a central unit (CU) and a distributed unit (DU), the latter can contain the radio unit (RU) or it can be a separate unit. RU is a specific hardware that uses radio resources to enable radio communication for UEs. The radio resource allocation within a single cell or multiple cells is done by schedulers.

Optimization on radio resource allocation is the most relevant research direction in RAN slicing as it affects the performance most. As parts of gNB are placed on different physical locations, the VNF placement optimization is one research direction. VNF placement contributes to the efficient use of computing, storage and networking resources available on different physical locations, called points of presence (PoP), where the resources are available. Resource allocation is an NP-hard optimization problem. It has been shown for computing, storage and networking resources in [10] and [11], and for radio resources in [12] and [13], among others. Slice scheduler allocates radio resources to slices, UE scheduler allocated radio resources for UEs according to the requirements of the corresponding slice. The goal is to accomplish efficient radio resource allocation which satisfy given requirements.

The VNF requirements can directly translate to the need of computing, storage and networking resources. The requested rate in RAN cannot be translated directly to the number of resource blocks (RBs). The achieved rate in RAN depends on the signal quality parameters achieved by UE, which is stochastic in its nature.

The slice has a life cycle specified in TS28.530 [14]. Life cycle management (LCM) is another research direction. When the slice is needed, then it is created. This can be triggered by the new slice request. During the slice operation, its performance is monitored. Slice monitoring can be done by using closed control loop. When the need to modify the slice is detected (e.g. resource utilization overload or underload), then the slice is modified. When the slice is not needed (e.g. no UEs left), it is deleted.

The VNF placement for efficient computing, storage and networking resource allocation, scheduling optimization for radio resource allocation and slice life cycle management are all different aspects of resource allocation optimization.

1.1 Motivation

The motivation of this thesis is the difficulty of answering the question of how many RAN slices can be present in the 5G network. In 5G standards there are specified three standard slice types for three verticals in Rel-15, fourth for vehicle to everything (V2X) in Rel-16, fifth for High-Performance Machine-type Communications (HMTC) in Rel-17 and sixth for High Data rate and Low Latency Communications (HDLLC) in Rel-18 [15]. Additionally, there is an option to create custom slices that can be used by MVNOs when their service requirements do not match any of the standardized slices.

To evaluate the possible number of slices in the RAN, the simulations were done where one RAN BWP was sliced into 2 to 10 slices. The performance results showed that the bandwidth utilization, network throughput and goodput and block error ratio (BLER) in both uplink (UL) and downlink (DL) were not constants. Results are shown in my publication 1 [1]. This motivated the investigation of how many RAN slices can be present from the performance perspective. This is investigated by subslicing the slice, which uses the RAN bandwidth part.

1.2 Problem Statement

The slice performance depends on how many subslices it contains; however, the slice performance dependence on the slice size is an open issue which needs to be investigated as a first step. If the slice performance dependence on the slice size can be formulated, then the next open issue to investigate is how this knowledge can be used to split the slice into suitably sized subslices to improve slice performance. Slice performance improvement means that bandwidth utilization has decreased, goodput per one RB increased and slice average UE BLER decreased. The next issue to investigate is how this knowledge can be used for slice performance improvement on fixed slice bandwidth; the last issue to investigate is how a management closed control loop can automatically detect the need for RAN subslicing, configure the subslices, and monitor the slices and subslices performance.

1.3 Research Questions

In this PhD thesis, I address the following research questions (RQ).

RQ1: how many subslices can be in the RAN slice? Standards do not specify the upper limit to the number of slices. Thus, it depends on the performance of the RAN.

RQ2: how much radio resource can be allocated to the slice and subslice to achieve the best performance? Resources must be sufficient to satisfy the requirements of the slice. However, when the RAN slice performance depends on the allocated BWP size, then it can be used to find best sizes for subslices, what slice can consist of.

RQ3: How users should be clustered, and slice bandwidth be allocated to subslices to improve RAN slice performance? RAN subslicing process includes division of UEs in the groups and allocate slice bandwidth subpartitions to the UE groups.

RQ4: What is the impact of RAN subslicing under the management closed control loop framework? Management closed control loop must detect the need for RAN subslicing, create configuration for subslices and monitor the performance of subslices and slice.

1.4 Contributions

The thesis contributes to advances towards RAN slicing, particularly analyzing how and when the performance can be improved while doing subslicing. Further, the thesis proposes several subslicing algorithms and finally proposes management closed control loop automation for RAN slice by subslicing.

I have simulated 5G-NR using MATLAB 5G Toolbox system-level simulation tool called NR Cell Performance Evaluation with Physical Layer Integration [16], version R2021b. Based on this simulated network, I investigate the RAN slice performance on fixed bandwidth constraint on RAN.

For RQ1, I discovered that if there are too many subslices, the slice performance in terms of high bandwidth utilization and low achieved goodput is poor because the subslices are too small. This is published as my publication 1 [1]. To improve slice performance, the subslice cannot be too small. This is the starting point for the rest of the research.

For RQ2, I simulated all possible sizes of subslices. For each RB allocated, I have a UE which requests the rates such that it is able to consume that RB. This gives the performance of one UE if it can use one RB in subslices at different sizes. Simulation results show subslice performance dependence on subslice size.

The RQ3 deals with subslicing algorithms. Subslicing contains 2 activities - group UEs and divide slice bandwidth. UEs can be grouped randomly or clustered using some algorithms by some feature. I propose UE clustering algorithm which can cluster UE into multiple clusters if the feature has single value. Moreover, the minimum subslice size limit is in bandwidth units, RBs. For creation of subslices in desired size, it must be converted into minimum cluster size. In my publication 2 [2] all UEs request the same rate, the minimum subslice size is converted to minimum cluster size before starting UE clustering. The proposed subslicing algorithm called user clustering with bandwidth allocation (UCwBA) creates given number of subslices with minimum subslice size not exceed the set constraint. In my publication 4 [4] the UEs request different rates, and minimum subslice size is converted to cluster sizes by calculation of cluster's sum rate. The proposed subslicing algorithm called dynamic subslicing by BLER (DSbB) splits slice or subslice into two and both splits are greater than set minimum subslice size requirement.

For RQ4 I implement known Monitor-Analyze-Plan-Knowledge (MAPE-K) closed control loop for introducing subslicing if it is able to improve slice performance. The closed control loop is published in my publication 4 [4]. For Plan function, the classifier neural network is trained to decide the operation "merge", "split" or "no change" operation for each subslice. The two methods to compose training data for neural network (NN) are proposed. The first method proposes performance data clustering in my publication 3 [3] and the second method proposes to solve optimization problem to find best subslice sizes in my publication 4 [4].

The first novelty of this work is the improvement in slice performance without additional bandwidth resources. Second, the subslicing has not implemented in the management closed control loop. Third, subslicing has been proposed to collect more similar UEs according to their requirements into one subslice, and the slice performance improvement has been noticed. This work proposes subslicing to collect similar UEs by their BLER into one subslice to improve slice performance.

1.5 Thesis Organization

The rest of the thesis is organized as follows (see Fig. 2):

- Chapter 2 provides the overview of RAN slicing, bandwidth allocation to slices and RAN slice management.
- Chapter 3 presents the slice performance dependence on slice BWP size. For each RB allocated, a UE is admitted, which is able to consume it by its requested rates. Two datasets, which are simulation results of subslices at all possible sizes in RBs are used as training data for deciding subslice operation in Chapter 5.
- Chapter 4 presents the proposed RAN subslicing algorithms, which can be used in Execute function of management closed control loop (MCCL) in Chapter 6.
- Chapter 5 presents the method to decide subslice operation for automatic subslice reconfiguration in Decide/Plan function of the MCCL implementation proposed in Chapter 6.
- Chapter 6 describes the MCCL implementation and evaluation.
- Chapter 7 concludes the thesis.

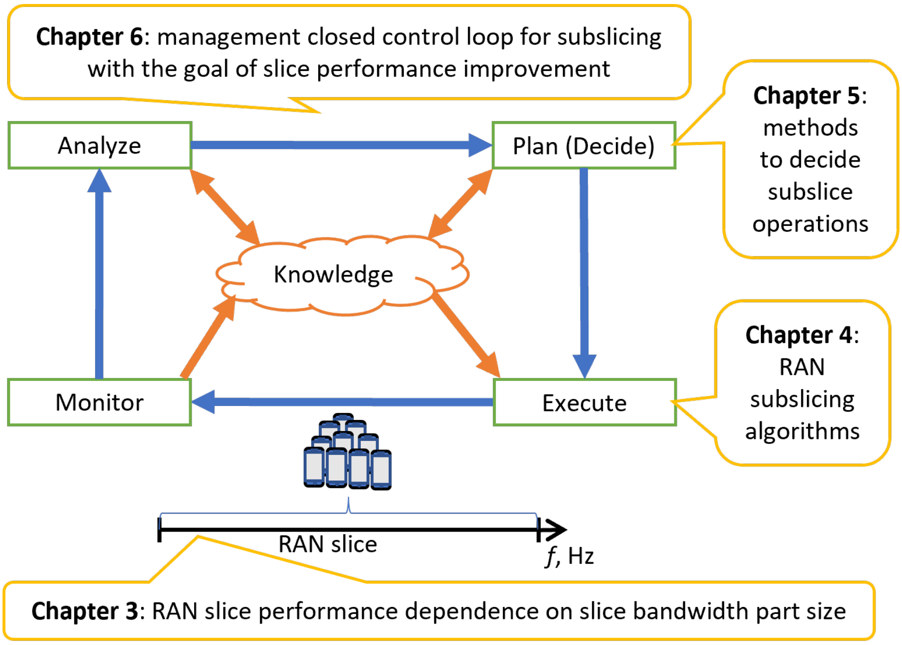

Chapter 6: management closed control loop for subslicing with the goal of slice performance improvement
Analyze
Plan (Decide)
Chapter 5: methods to decide subslice operations
Knowledge
Monitor
Execute
Chapter 4: RAN subslicing algorithms
RAN slice
f, Hz
Chapter 3: RAN slice performance dependence on slice bandwidth part size

Figure 2: MCCL and thesis organization.

2 Background and State of the Art

In this chapter, I begin by providing an overview of network slicing, followed by an exploration of RAN slicing specifics and RAN resource allocation. I then explain the slice lifecycle, followed by the overview of subslicing. Finally, I discuss the capabilities of 5G-NR simulators and emulators in supporting RAN slicing.

2.1 Network Slicing

Network slicing enables to create multiple logical networks, called slices, on top of physical infrastructure that allows a 5G network to be partitioned into multiple logical and independent networks, each optimized to serve a particular use case. 5G system consists of core and RAN, while the transport network (TN) provides the connectivity. The network slice consists of core slice and RAN slice, and resource slices in other domains if needed.

2.1.1 Core Slicing

The 5G core network has a service based architecture. The network service is realized using network functions. The network service consists of virtual network functions (VNFs), VNF Forwarding Graphs and Virtual Links. VNF Forwarding Graph represents the topology of the network service, i.e. connections of constituent VNFs used in the network service. Virtual Links define requirements for connections specified in VNF Forwarding Graph. The VNFs need computing, storage and networking resources to run on. The VNF runs on a Network Function Virtualization Infrastructure (NFVI) node, whose architecture is specified in ETSI NFV-EVE 003 [17]. NFVI node consists of computing, storage and network nodes. The resource management and orchestration (MANO) mechanism (specified in ETSI NFV-MAN 001 [18]) is used to manage the dependencies of resource nodes and their placements. On the physical infrastructure, the Data Centers (DC) are located in different geographical locations. DC contains racks with provided power and cooling to the resource devices. The connectivity inside DC is provided by switches and to other DCs by gateways. The resource devices are computing hardware (CPUs, etc with network interface) and storage hardware (hard disks, Direct Attached Storage (DAS), Network Attached Storage (NAS), or Storage Area Networked (SAN)). Further information can be obtained from ETSI GS NFV-INF 003 [19]. Research areas in core slicing include efficient utilization of computing, storage and networking resources and VNF placement, among others.

TN slicing is implemented using software-defined networking (SDN) by defining links and their capacities and requirements. Edge resources are incorporated as an edge slice subnet if services need edge access. Without SDN, the network provides best-effort data transmission. SDN decouples the control and data planes. A controller implemented in the software controls all physical routers and switches. It is possible to specify certain quality of service (QoS) requirements for packet flows specified by their criteria, for example, source or destination address and contained protocols. One example SDN protocol is OpenFlow [20].

2.1.2 RAN Slicing

The RAN slice is defined as a network slice subnet in RAN domain. Network slice subnet contains NFs and resources. The core network primarily comprises NFs that are hardware-agnostic, allowing for the core to be composed solely of VNFs. RAN has specific hardware - antennas that use physical resource - frequency bandwidth. Therefore, RAN slice subnet must contain both VNFs and physical network functions (PNF).

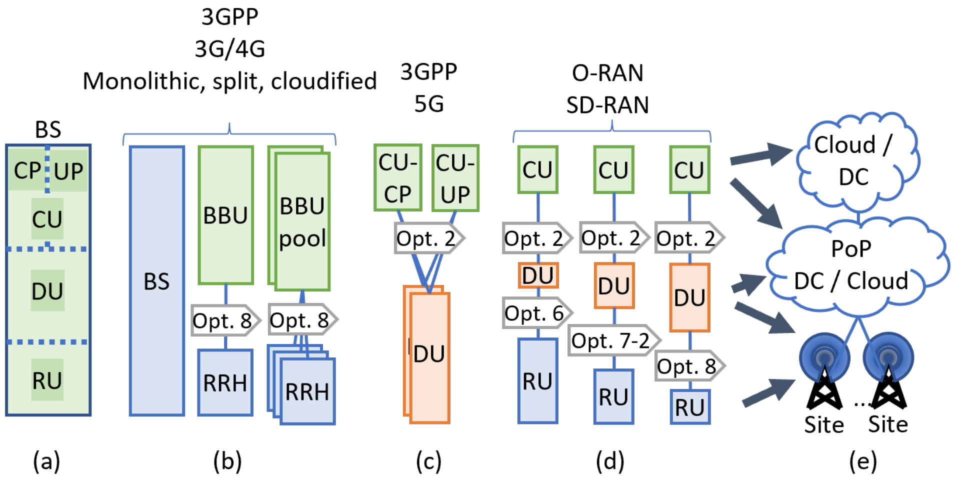


Figure 3: Base station decomposition - CUPS and functional splits. (a) possible decomposition of a BS, (b) BS split and cloudifying by 3GPP and used in 3G/4G RAN, (c) CUPS and CU-DU split used in 5G RAN, (d) SD-RAN split into 3 units by O-RAN, (e) possible physical placement of BS components

The decomposition of a BS enables to place some of its higher level functions implemented as VNFs into DC or cloud to increase the reliability and security. The DCs have better power and connectivity, and cloud has better network accessibility. The BS decomposition options are specified in 3GPP TR 38.801 [21] and are shown in Fig. 3a. The BS can be split horizontally into up to three units - central unit (CU), distributed unit (DU) and radio unit (RU). If BS is split into two units, then DU and RU are together in one DU. There exist 8 horizontal (functional) split options. The CU can be further split vertically into control plane and user plane, referred to as control plane and user plane separation (CUPS). This enables to reduce UE delay by placing some components of gNB geographically closer to UE and to provide better reliability to place some BS components into data center providing higher capacity.

The Fig. 3b illustrates used split options starting from 3rd generation (3G) cellular communication when the BS was split into Base Band Unit (BBU) for computing tasks and Remote Radio Head (RRH) that needs to be placed next to the antenna. The Cloud/Centralized RAN (C-RAN) uses BBUs in cloud environment and this solution is used in 4G as well. In Fig. 3c the split of 5G BS is presented. 5G as defined by 3GPP uses CUPS and CU-DU split option 2 (pass RLC channels). The O-RAN's Software Defined RAN (SD-RAN) splits gNB into 3 parts - CU, DU and RU as it is displayed in Fig. 3d. In O-RAN solution there exist 3 different splits between DU and RU - option 6 (pass physical channels), option 7-2 (pass antenna symbols) and option 8 (pass in-phase and quadrature (IQ) data, a complex signal representation). O-RAN specifies open interfaces between CU, DU and RU to enable interoperability between devices from different vendors [22]. The RAN intelligent controller (RIC) controls the RAN nodes. The Fig. 3e shows possible physical placements of different units of RAN base station. The PoP DC is located closer to the antenna site for shorter delay of user plane traffic. This enables to find optimal physical placement of RAN VNFs for different slices.

The network node in RAN is a base station (BS), called gNB in 5G. The virtualization and splitting of BS, enables to place computing nodes into geographical locations where it is more convenient to provide a power, storage, security and have only minimal needed computing tasks near the antenna's physical location. Slicing the RAN enables handling of different QoS Flows by different RAN architecture elements.

The upper protocol layers of gNB i.e. gNB-CU can be implemented as VNFs with virtual resources and lower layer protocols down to physical antenna are implemented using PNFs with physical resources such as frequency bandwidth to transmit and receive UE data. The connections between VNFs and PNFs are done using TN with specified TN requirements. The lower protocol layers of gNB i.e. gNB-RU or gNB-DU depend on physical hardware and must be implemented as PNFs.

There are no VNFs defined for RAN, and RAN is not required to be virtualized by 3GPP standards. The main idea of virtualization of gNB (base station) is to separate (disaggregate, split) computing and storage tasks from radio transmitting and receiving tasks and place the antenna hardware with minimum required computing and storage in the necessary geographic location.

The O-RAN alliance has the RAN virtualized. The Near-Real Time RAN Intelligent Controller (RIC) enables control and optimization of all gNB units via interface E2 [23].

2.1.3 Network Slice Template

GSM Association (GSMA) has defined a Generic Network Slice Template (GST). Versions 1.0 and 2.0 [24] dated 2019 are compatible with 3GPP Rel-15. Versions 3-9 (v9 [25] are compatible with 3GPP Rel-16.

In the template, various slice attributes are defined. The attributes have the following parameters defined: name, description, typical values, parameters (parameter unit, allowed values, tags) and attribute presence (mandatory, conditional, optional). GST Attribute categories are: character attributes (to characterize a slice), scalability attributes (about the scalability of the slice), performance related (KPIs supported by a slice), function related (functionality provided by slice), control and management related (which methods are provided for the network slice consumer in order to control and manage the slice). NEtwork Slice Type (NEST) is a GST filled with values.

Usually, UEs with similar requirements can work in the same slice. A new feature introduced in GST v5.0 allows defining some slice attributes for different service categories within the slice, i.e., UEs that operate in the same slice and belong to different service categories can have different requirements related to uplink (UL) and downlink (DL) maximum UE throughput and supported data network. In addition, by using slice attributes it can be defined whether the network is simultaneously used by other slices. The combination of these attributes enables to define one slice for each MVNO with different per UE QoS requirements in one slice.

2.1.4 Network Slice Identifier

The network slice identifier is called Single Network Slice Selection Assistance Information (S-NSSAI) and it consists of two fields (3GPP TS 23.501 [15], 3GPP TS 38.300 [26]):

- mandatory Slice/Service Type (SST), field length is 8 bits (256 different values)

- optional Slice Descriptor (SD), field length is 24 bits (16777216 different values). If SD does not exist, then SD has a value of FFFFFF (TS 23.003 [27]).

The standardized SST values are defined in 3GPP TS 23.501 [15]:

- SST=1 enhanced Mobile Broadband, eMBB
- SST=2 massive Internet of Things, MIoT (to serve mMTC vertical)
- SST=3 Ultra Reliable Low Latency Communication, URLLC
- SST=4 Vehicle to everything, V2X (added in Rel-16)
- SST=5 High-Performance Machine-type Communications (HMTC) (added in Rel-17)

The Network Slice Selection Function (NSSF) is a dedicated function to select the appropriate slices (3GPP TS 23.501 [15], TS 29.531 [28]) using network slice selection policy (NSSP). The information about all slices is called Network Slice Selection Assistance Information (NSSAI) that consists of up to 8 S-NSSAI-s, i.e., slice identifiers. A UE is able to use at most 8 slices simultaneously. The proper slice is selected in process of *RRCSetupComplete*. If UE does not request the slice, then it will be connected to the default Access and Mobility Management Function (AMF), otherwise it will be connected to the AMF that supports the requested slice.

If a non-standard S-NSSAI is used, then this S-NSSAI is used only on the Public Land Mobile Network (PLMN) to which it is associated. Slicing supports roaming in case standard values for SST and optionally SD are used. Then the same values can be used in Visited PLMN otherwise it is complicated - AMF selects suitable numbers for slice identifier or with the help of other core functions the proper network functions are mapped to the visited slice (3GPP TS 23.501 [15]).

2.1.5 Management of Slices

Network management in general is Fault, Configuration, Accounting, Performance and Security (FCAPS) management. In addition, a lifecycle of a slice is also managed.

There are 3 management functions defined in 3GPP TS 28.533 [29] and ETSI GR NFV-EVE 012 [30]:

- Communication Service Management Function (CSMF) - translates and maps service and slice related requirements
- Network Slice Management Function (NSMF) - manages Network Slice Instances (NSI) and their lifecycle
- Network Slice Subnet Management Function (NSSMF) - manages Network Slice Subnet Instances (NSSI) and their lifecycle. There are separate NSSMFs for core and RAN.

The lifecycle of NSI is defined in TS 28.530 [14]. The example of life cycle management (LCM) of a slice with one subnet in CN and one subnet in RAN is shown in Fig. 4. First, the SLA between slice provider and slice consumer is agreed. Then agreed slices are designed, onboarded by CSMF and the network environment is prepared to use the slice. The result of preparation is that the system has a database (DB) of possible slices and slice constituents. Now, when the slice is needed to work, then the slice lifecycle begins with creation of NSI that is put together from constituent NSSIs by NSMF and NSSMFs at each domain. Next, the NSI and constituent NSSIs are activated. During the operation phase, NSI can be modified according to the requirements and resource usage. The NSI modification means that the constituent NSSI is modified; NSI or NSSI reconfiguration means that NSI or NSSI as a logical network has been reconfigured, NSI or NSSI scaling means that the resources can be added (scale up) or removed (scale down) from the NSI or NSSI

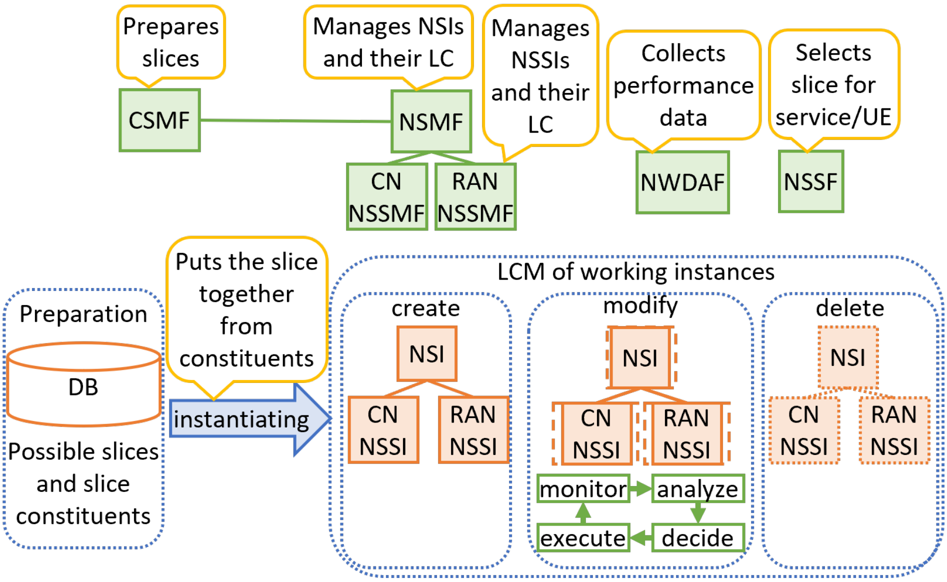

Prepares slices
Manages NSIs and their LC
Manages NSSIs and their LC
Collects performance data
Selects slice for service/UE
CSMF
NSMF
CN NSSMF
RAN NSSMF
NWDAF
NSSF
Puts the slice together from constituents
Preparation
DB
Possible slices and slice constituents
instantiating
LCM of working instances
create
modify
delete
NSI
CN NSSI
RAN NSSI
monitor
analyze
decide
execute

Figure 4: Slice LCM example in case of slice consists of one CN slice subnet and one RAN slice subnet.

respectively. The data about slice performance and overall performance from NetWork Data Analytics Function (NWDAF) can be collected for deciding if slice modification is necessary. The closed control loop (CCL) with monitor, analyze, decide and execute functions, defined in 3GPP TS 28.535 [31] can be used to monitor the performance and trigger NSI or NSSI modification. When the working slice instance is not needed, no UE uses it or other conditions are true, the NSI can be deleted (deactivated and terminated). For dynamic slicing, the slice instances can be created, modified and deleted automatically.

According to 3GPP TS 28.541 [32] the states of (NSI and) NSSI are:

- operational state (Enabled/Disabled) - if physically installed and working;
- usage state (Idle/Active/Busy) - if actively in use and if so then if it has spare capacity;
- administrative state (Locked/Unlocked/Shutting down) - describes the permission to use.

Network slice subnets are managed in collaboration with 3GPP Management System and ETSI NFV-MANO: 3GPP Management System manages PNFs and ETSI NFV-MANO manages VNFs. 3GPP management system manages NFVI collaboratively with NFV-MANO as defined in 3GPP TS 28.500, [33]:

- Fault Management: 3GPP management system receives NFVI and VNF and virtualized resource alarms (type, severity, possible cause), from NFV-MANO. 3GPP can request NFV-MANO to recover VNF.
- Configuration Management: 3GPP management system manages the configuration of physical and virtual 3GPP entities (with objects and attributes), and reconfigures physical and virtual entities (and neighbor 3GPP entities) due to LCM operation of VNF or service from NFV-MANO. VNF instance scaling will not modify existing connections with other network entities. 3GPP management system is able to request NFV-MANO to modify VNF instance managed by NFV-MANO.

- Performance Management: KPIs are applicable for both physical and virtualized entities. 3GPP management system supports performance measurements and results for virtualized entity and virtualized resources. NFV-MANO provides VNF PM data to 3GPP management system, which can request VNF LCM operation to mitigate VNF performance bottleneck.

- Life Cycle Management: 3GPP management system can request the service and VNF LCM operations, and service, service descriptor and VNF management operation and VNF Healing from NFV-MANO. NFV-MANO sends notifications to 3GPP management system about service, service descriptor and VNF management and VNF LCM change. 3GPP management system can enable or disable autoscaling of corresponding VNF instances in NFV-MANO.

NFV-MANO has standards to specify information model, NFV infrastructure, components of service templates, and interfaces. Network functions are virtualized using virtual machines or container platform.

OSM is an open source NFV-MANO software stack development using open source tools and working procedures. There exists ETSI OSM group and the OSM software is hosted by ETSI [34].

2.1.6 Management of Resources

The resources are computing, storage and networking resource. Resources can be physical, logical or virtualized. Logical resources are not limited to physical resources, virtualized resources have mapping to physical resources.

In RAN, there are radio resources - bandwidth. The radio frequency (RF) spectrum is divided into frequency ranges (FR). 5G uses 2 FRs: FR1 is 450-7125 MHz and FR2 is 24250-52600 MHz [9]. FR contains operating bands, denoted as n1 to n100. Operating bands contain bandwidth parts (BWPs) that can be assigned to a slice with GST. BWP contains a number of Physical Resource Blocks (PRB). The 3GPP TS 38.211 [35] explains the PRB size in frequency and time scales.

The constituents of Resource Block (RB) in the time and frequency scale are shown in Fig. 5. One PRB contains 12 subcarriers that are resource elements on the frequency dimension. On the frequency dimension, the mapping of PRB to a frequency unit depends on subcarrier spacing (SCS) configuration, denoted as $\mu \in \{0,1,2,3,4\}$. The SCS can be calculated as follows: $\Delta f = 2^{\mu} \cdot 15$ [kHz]. The stated length on time dimension is a frame duration that is 10 ms and can be divided into 2 half frames. The frame contains 10 subframes. There are 2 slots in one subframe. The slot contains 12 or 14 Orthogonal Frequency Division Multiplexing (OFDM) symbols if extended or normal cyclic prefix is used accordingly. The duration of a slot can be calculated from duration of a subframe (1 ms) and a count of slots in subframe, that depends on SCS configuration μ and can be calculated as $N_{slot}^{subframe,\mu} = 2^{\mu}$.

The RAN-specific resource to be managed in the RAN is the RF spectrum. Spectrum sharing means that the same frequencies can be used by many slices but at different times. Resource allocation determines the time at which frequencies are used by the slice. Resource isolation means that the frequency used at this time by this slice does not interfere with other slices, and other slices do not interfere with this slice.

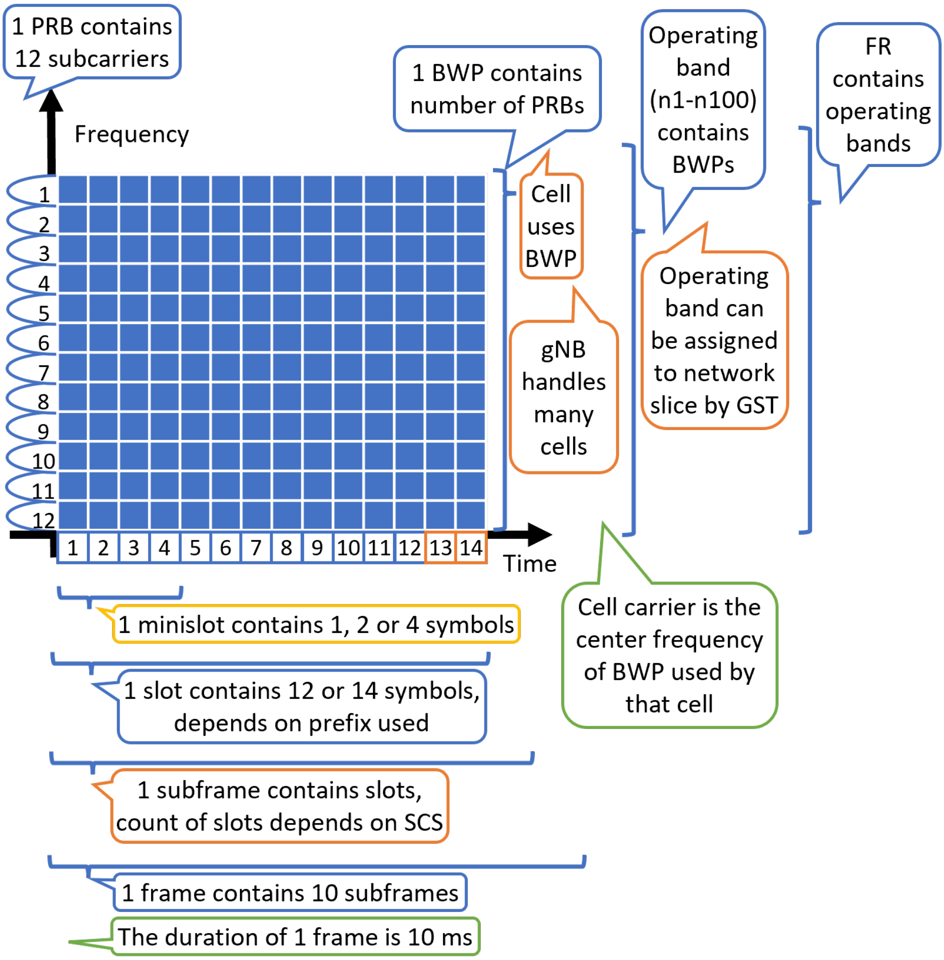

1 PRB contains 12 subcarriers
Frequency
1 BWP contains number of PRBs
Operating band (n1-n100) contains BWPs
FR contains operating bands
Cell uses BWP
gNB handles many cells
Operating band can be assigned to network slice by GST
1
2
3
4
5
6
7
8
9
10
11
12
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Time
Cell carrier is the center frequency of BWP used by that cell
1 minislot contains 1, 2 or 4 symbols
1 slot contains 12 or 14 symbols, depends on prefix used
1 subframe contains slots, count of slots depends on SCS
1 frame contains 10 subframes
The duration of 1 frame is 10 ms

Figure 5: The constituents of resource block according to 3GPP TS 38.211 [35]. On the time dimension, it contains the 12 or 14 symbols that form a slot. On the frequency dimension, it contains 12 subcarriers that form a resource block.

2.1.7 Automation in Management

The management closed control loop (CCL) described in 3GPP TS 28.535 [31] and ETSI ZSM 009-1 [36] is used for automation in the management of resources used by the communication system. An overview of the control loop architectures can be found in ETSI GR ENI 017 [37]. The control loops are used for adaptive and cognitive systems which adapt to the recognized changes in their environment.

Examples of well known CCLs are MAPE-K (Monitor-Analyze-Plan-Execute, Knowledge) and OODA (Observe-Orient- Decide-Act) [38].

One implementation of CCL is in [39]. UEs are ships in real-world seaport testbed. The signal quality parameters are monitored along with UE movement. When UE is behind the large objects, the signal quality deteriorates more than its stochastic variance. The prediction of UE movement enables to predict its signal conditions for longer term, which enables the large-scale network reconfiguration. When the radio link failure predicted, then the beam adjustment of its power and tilting is performed to avoid the radio link failure.

In slice resource management, the CCL works as follows: the slice load is monitored. If slice overload occurs, then additional resources are allocated. If slice underload occurs, then some resources are removed from the slice.

2.1.8 Summary

3GPP has defined the 5GS and support of network slicing in RAN. ETSI has defined NFV-MANO system to manage network services. 3GPP has defined the management of network slices that provide logical network to run for services or tenants. 3GPP management system uses ETSI's NFV-MANO to manage VNFs. Both 3GPP and ETSI management systems support the slice template that is defined by GSMA. O-RAN Alliance has developed its own solution of SD-RAN that uses 3GPP-defined RAN and has some additional features.

2.2 RAN Schedulers (Radio Resource Allocation)

The 3GPP TS 38.300 [26] states that the scheduler assigns resources between UEs by considering the UE buffer status, QoS requirements, and associated radio bearers. In addition, the scheduler may consider UE radio conditions measured at the gNB and/or reported by the UE. Measurements include uplink buffer status for UE to provide QoS-aware packet scheduling and power headroom report (measured difference between UE maximum transmit power and estimated uplink transmission power) to provide power-aware packet scheduling. In LTE QoS-based schedulers exist that can schedule UEs depending on QoS requirement in uplink [40] and in downlink [41].

Traditional schedulers must schedule UEs to use the available infrastructure. Without slicing, all UEs are equal with equal SLAs, or there can be UEs with different values of one QoS parameter in the SLA relevant to the scheduler. With slicing, schedulers need to schedule slices to use infrastructure resources (inter-slice scheduling) and schedule UEs to use slice resources (intra-slice scheduling). In addition, slices and UEs can have different types of requirements, for example, both throughput and latency.

The information about schedulers is in Tab. 2.

There are three types of scheduling approach for RAN slicing:

- Inter-slice schedulers - schedule slices to use infrastructure resources
- UE schedulers - schedule UEs in slices to use infrastructure resources or slices resources
- Puncturing - allocate one RB for two UEs belong to different slices

A simplified scheme of inter- and intra-slice schedulers for latency, fixed, throughput, and best-effort slices is shown in Fig. 6. The scheduling process is illustrated from left to right. The packets are labeled as follows: letters as different slices (B - best-effort slice; L - latency-constrained slice; T - throughput-oriented slice; M - uses fixed resources) and numbers as different UEs in that slice. On the left side, the packets arrive. At the inter-slice scheduler, the queues for each slice are formed, and the L-slice sends its packets with priority. Flexible resources are allocated to the L-slice according to the queue length during the cycle. Then, the T-slice can have the resources that it needs. The remaining resources are allocated to the B-slice. The M-slice uses a fixed resource count for each cycle. On the right-hand side, a simplified resource grid with resources allocated to slices is shown.

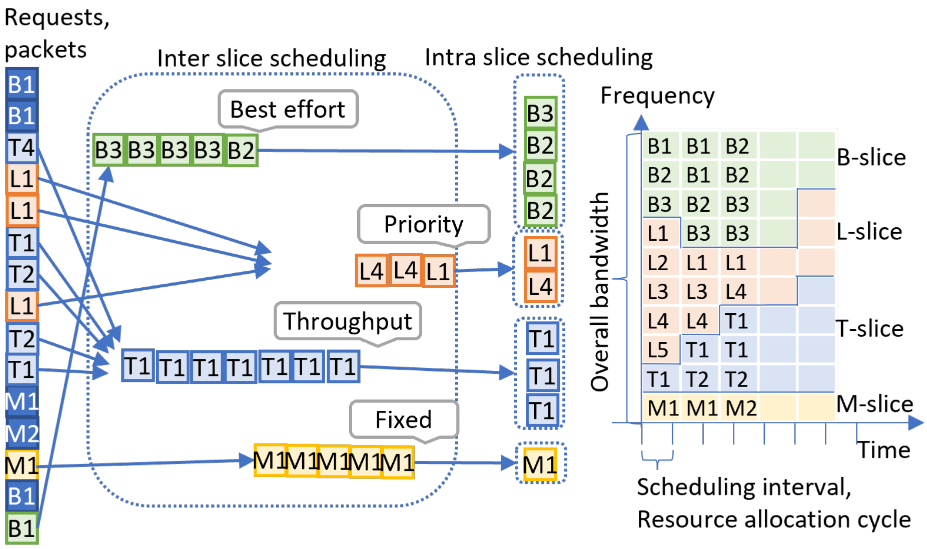


Figure 6: Simplified scheme of inter- and intra-slice scheduling of latency (L-slice), fixed (M-slice), throughput (T-slice) and best effort (B-slice). Request or packet notation contains a letter to identify the slice and number to identify UE in that slice. The figure shows a snapshot at a given time instant.

Slicing adds another layer of complexity and requirements from the scheduler; thus, a scheduler should be able to fulfil certain requirements in terms of scheduling level, domain, checked SLA, scope, and complexity. The complete scheduler for the RAN slicing is defined as follows:

- **Scheduling level:** The complete scheduler can schedule both slices to use the infrastructure and the UEs in the slices. It can be modular, such that one is an inter-slice scheduler, and in each slice, a different slice-specific intra-slice scheduler is working. The other option is that it is a UE scheduler, for example, for MVNO slices that schedules UEs with each UE having a different SLA in terms of rate, latency, and reliability.

- **Domain:** The complete scheduler works in both the time and frequency domains and manages the transmission power.

- **SLA checked:** The complete scheduler considers the rate, latency, and reliability requirements from slice SLA.

- **Optimization goal:** All slice and UE SLAs are satisfied with efficient bandwidth and energy utilization.

- **Scope:** One or multiple cells. To minimize inter-cell interference, the scope of multiple cells is required.

- **Complexity:** As the optimization problem is NP-hard [12], [13], the complexity should be as low as possible to achieve sufficient results for the optimization goal. The time required to solve the optimization problem must be less than the length of the scheduling interval.

- **When there are more requests than available resources can serve:** more stricter SLAs are satisfied, and others are served with fairness.

In the literature, the proposed schedulers do not satisfy all the aforementioned requirements and capabilities. Hence, we compare the schedulers and list their limitations or completeness in terms of what a complete RAN slicing scheduler should be able to do.

2.2.1 Schedulers comparison

The schedulers described in this section are selected based on the allocation of radio resources and the satisfaction of multiple requirements. The schedulers are compared in Tab. 2. The inter-slice schedulers are [42], [43] and [44]. All of these schedulers allocate RBs to slices according to rate and latency or delay. Using inter-slice schedulers, separate intra-slice schedulers can be used in each slice. Both inter- and intra-slice schedulers are proposed in [45], [46], [47], [48] and [49]. These schedulers schedule both the slices to infrastructure and UEs to slice resources. These have the most complex optimization goal, which is usually different for each slice (UE group). This motivates the scheduler architecture to be multilevel or hierarchical: first, schedule slices with stricter or demanding requirements and then others, or first schedule slices and then UEs in their corresponding slice. In [48] and [49], both inter- and intra-slice scheduling decisions are learned using DRL and multi-agent RL, respectively. UE schedulers are proposed in [50], [51], [52] and [53]. The UE scheduler, which schedules UEs in different slices, considers the UE QoS requirements regarding its slice. RBs are allocated to the UEs with UE or BS transmit power management.

Most schedulers used for LTE enable one QoS requirement to be satisfied with different values, either the data rate or latency [41]. The schedulers used for RAN slicing must satisfy different QoS requirements, such as throughput and latency at once. In addition, more QoS requirements should be satisfied, as the slice SLA defines several QoS requirements. The eMBB slices are known to have high-throughput requirements. The achieved throughput depends on the number of allocated RBs and signal quality. URLLC slices are known to have low latency requirements. Latency depends on the amount of time the packet spends on the transmission queues, and this time can be reduced using proper scheduling algorithms. The parameters for slice traffic QoS requirements can be extracted from 5G QoS Identifier (5QI) and other QoS flow parameters to obtain the data rate, latency, priority, and packet error rate requirements [54]. Slices with low delay and latency requirements are handled by a scheduler with priority, and then other slices and slice UEs are scheduled. In [42] first URLLC slice is scheduled, and in [43] a slice with fixed resources, an on-demand (URLLC) slice, and a third dynamic (eMBB) slice is scheduled. In [46] the URLLC slice that has UEs that require radio resources for communication with gNB and D2D UEs is scheduled first. MVNO slices have UEs with different QoS requirements in the same slice.

Time domain scheduling is the allocation of time to satisfy latency and priority requirements. The slices/UEs with latency, delay, and priority requirements (e.g., URLLC slice) should be scheduled first or have less waiting time in the queue. The earliest deadline first (EDF) scheduler [55] prioritizes UEs which required delay ends sooner. With slicing, the URLLC slice has priority over others in [46], or in [43] fixed slices are first followed by URLLC slices and then others.

Frequency domain scheduling is the allocation of frequency to satisfy the throughput and data rate requirements in the SLA. The achievable throughput is usually calculated using Shannon's formula [56] from the bandwidth and signal-to-noise ratio (SNR) or signal-to-interference-plus-noise ratio (SINR). The throughput achieved depends on the

Table 2: Schedulers comparison

Paper, year	**Scheduling level**	**Traffic direction**	**Transmit power management**	**SLA QoS requirements to be satisfied**	**Scope**	**Optimization goal**
[42], 2019	Inter-slice	DL	No	Rate and delay	Single gNB	Maximize rate
[43], 2019	Inter-slice	DL	No	Rate and delay	One BS (cell)	Efficient resource usage
[13], 2019	Inter-slice	DL	Yes	Slice throughput	Multiple cells	Reduce inter-MVNO interference
[44], 2020	Inter-slice	DL	Yes	UE rate and UE latency	Multiple cells	Maximize throughput and minimize delay
[45], 2019	Both	DL	No	Slice latency and throughput	Multiple cells	URLLC latency below threshold
[46], 2020	Both	UL and DL	Yes	Rate, delay, packet loss, connection density	Multiple cells	URLLC: minimize delay, maximize average rate; eMBB: maximize slice energy efficiency; mIoT: minimize average power
[47], 2022	Both	DL	No	URLLC: packet loss, delay; eMBB: average throughput	Multiple cells	With URLLC reliability guaranteed minimize cost
[48], 2021	Both	DL	Yes	General slice QoS	Single cell	Maximize QoS and spectral efficiency
[49], 2021	Both	Not specified	No	UE delay and throughput	Multiple cells	URLLC: minimize delay; eMBB: maximize throughput
[50], 2020	UE scheduler	UL	Yes	Rate and delay	One BS (cell)	Maximize rate with delay constraint
[51], 2020	UE scheduler	UL	Yes	Rate and latency	One BS (cell)	Maximize overall system capacity
[52], 2020	UE scheduler	DL	Yes	URLLC: delay; eMBB: throughput	Single cell	Maximize URLLC reliability and network spectral efficiency
[53], 2020	UE scheduler	DL	No	URLLC: priority; all: bitrate and error rate	Single cell	Maximize network sum rate

number of bits transmitted/received within one PRB and the number of PRBs allocated. The number of bits per PRB that can be transmitted depends on the selected modulation and coding scheme (MCS), which in turn depends on the signal parameters, particularly the SNR or SINR. The bandwidth required to satisfy the data rate requirement is primarily calculated from the measured SINR or SNR. An example of a scheduler is presented in [45], where allocated RBs are adjusted later using UE CQI and MCS values.

All schedulers discussed allocate RBs, that is, work in both the time and frequency domains. Power management is closely related to the energy efficiency and the number of bits that can be transmitted within a PRB. If the SINR is too low and a power headroom exists, the transmit power can be increased; thus, a better SINR can be obtained, a higher MCS can be used, and more bits can be transmitted within one PRB. If the signal quality is good, then the transmission power can be reduced.

Schedulers for DL with transmit power management for BS are proposed in [13], [44], [48], [52]. In [13] the BS transmission power is reduced when different slices use matching RBs on adjacent cells. In [44] power management is in the optimization problem to calculate the power allocation factor for each slice as a fraction of the available power on the BS. Their utility maximization problem is formulated using stochastic geometry and Ljapunov optimization. The simulation results show that a tradeoff exists between throughput and delay in the slices. In [48] DRL is used to optimize the RAN configuration, including the optimal transmit power on the BS towards each active UE for the RB. Similarly, in [52] transmit power to the UE is calculated using joint subcarrier allocation as a result of solving the optimization problem. In [53] the transmit power of the BS is equally shared over RBs and used to evaluate the achievable rate via evaluation of the SNR and MCS.

Schedulers for UL with transmit power management for UEs are proposed in [50] and [51]. In [50] the UEs use eMBB and URLLC slices, the latter for gNB and D2D communication. Rate maximization with a delay constraint is formulated as a constrained MDP and relaxed using the equivalent Bellmann equation. The results demonstrate the ability to converge and improve UE performance. In [51] MINL stochastic optimization problem is solved using Lyapunov optimization. An asymptotically optimal resource allocation improves network capacity. A trade-off exists between the capacity and latency.

The optimization goal is to satisfy all SLAs with efficient resource usage. When radio resources are insufficient to satisfy all requirements, the slices or UEs should be selected, which does not have strict requirements and can be left unsatisfied. In this case, the service reliability or UE drop-rate values in the slice template are used.

SLA satisfaction with efficient resource usage is achieved by solving the optimization problem. Latency is attempted to guarantee using priorities and by solving the optimization problem to minimize delay. In [45] the optimization goal is to maintain the latency below the set threshold. The other optimization goal is to maximize the rate [42], [53], throughput, or capacity [51]. Rate or throughput maximization is achieved by selecting suitable RBs for UEs that can achieve maximum throughput. Different slices have different requirements, thus minimizing the delay for one slice and maximizing throughput for the other slice, which are defined as constraints of the optimization problem to maximize utility [44]. Multiagent RL is used in [49] to find the equilibrium of RB allocation to minimize the delay for URLLC slice and maximize the throughput for eMBB slice.

Efficient resource usage is the optimization goal in [43], QoS and spectral efficiency in [48], and URLLC reliability and spectral efficiency in [52]. In [52] the spectral efficiency maximization problem is relaxed to a convex optimization problem and solved by applying the Powell-Hestens-Rockafellar and branch-and-bound methods. Their simulation results show that the eMBB and URLLC slice requirements are satisfied, and the spectral efficiency improves. In [43] the dynamic slices have requested rate and requested guaranteed rate requirements. The first is served when more resources are available in the resource-allocation cycle. This enables to serve more slices.

In [13] the number of RBs to be allocated to MVNOs has already been determined, and the proposed algorithms determine which RBs are allocated to which MVNO to minimize inter-cell interference. To increase the throughput in the slice, multiple RBs must be linked (adjacent to each other), and the same RBs must be assigned to the same slice at different Base Stations (BS) to reduce the interference between signals belonging to different slices.

The situation in which the resources may not be sufficient to serve all slices are considered in [53] with the following assumptions: URLLC UEs create bursts of small packets and eMBB UEs create continuous traffic. Their proposed scheduler handles URLLC traffic with priority and a constraint that at least one RB for each scheduling period is scheduled for URLLC traffic to transmit at least one complete data packet. Each scheduled eMBB UE can transmit the number of bits in each frame.

The resources to be scheduled can be located on a single cell or multiple cells/BSs. In an unsliced network, there is a relationship between UE and gNB. The gNB serves cells and the UE connects to the cell with the strongest signal. Resources of gNB are allocated to the UE. This relationship is more complex in slicing situations. The gNB serves slices and cells. A slice can be served by several gNBs in several cells. The slice uses the fraction of the cell bandwidth. The UE can connect to the cell with the strongest signal but only if a suitable slice for the UE is served by that cell. Therefore, after UE admission, the cell bandwidth for the UE and the slice must be managed. The optimization over multiple cells require cell cooperation.

2.2.2 Summary

The schedulers are capable of efficient radio resource allocation within a specific standardized slice and a slice that serves the UEs with different QoS requirements. The schedulers for RAN slicing can satisfy one to four different QoS requirements, and the number of optimization goals are one or two. Efficient bandwidth allocation indicates that the slice is most of the time in the close-to-slice overload condition. In addition, if the bandwidth can be allocated by the granularity of the RB to the slice, the slice will always be in a close-to-slice overload condition.

2.3 Slice Life Cycle (Slice Provision)

2.3.1 Slice preparation and creation (provision)

First, the SLA between the slice provider and slice consumer is agreed upon. Then, the agreed slices are designed and onboarded, and the network environment is prepared for the slice. The result of the preparation is that the system contains a database (DB) of possible slices and slice constituents.

After the preparation phase, the work with slice instances is initiated. These working slices are automatically created, modified, and deleted according to the need and specified lifecycle of a slice instance. The architecture required for implementing the automatic lifecycle of a slice instance is described in [57]. The RAN slice subnet has two types of resources: physical and virtual resources. The physical resources considered are specific hardware and frequency bandwidth. A specific hardware can use a specific frequency bandwidth. A promising solution is implemented with points of presences (PoPs). PoP is a geographical location that contains data centers and antennas that enable the use of different frequencies to serve UEs in selected geographical areas. The slices are created or modified by deploying gNB VNFs to work using resources at DCs on PoPs and existing gNB-RUs as PNFs in cell site PoPs. Cells use fixed BWPs that can be switched on or off as a result of the slice LCM operation. For each cell, the slices it serves, and the fraction of radio resources allowed for each slice is set up.

At the slice creation context, the slice provisioning is to provide a new slice if a new slice is requested from slice provider. Slice provisioning is joint slice admission and slice resource allocation on scheduling. This ensures that the SLA is guaranteed. The QoS provisioning scheduler is only possible if slice admission is implemented [41].

In the slice preparation phase, a slice is described in the system. The slice design platform demonstrated in [58] functions as part of a slice orchestrator. The [59] proposes a slice design scheme that consists of slice descriptors such as slice size (count of RBs) to meet the data rate requirement and slice shape (count of consecutive RBs to be allocated) to meet the latency requirement of a slice. The traffic types considered are deterministic aperiodic, deterministic periodic, and non-deterministic traffic.

To decide whether to create a new slice or scale up an existing slice, [60] proposed a service admission algorithm that compares the cost of creating a new slice for the service with the cost of using the existing slice for the service. First, the cost of satisfying the requirements of a new slice is calculated. The cost is then calculated for each slice if it is extended to fit the new requirements to the existing slice. The minimum cost determines whether a new request is satisfied by the creation of a new slice or by using an existing slice. If the operator-defined cost threshold is smaller than the slice cost, then a new service is not admitted to the network.

AI/ML is used to predict resource requirements [61] or traffic [62]. In [61], big data analytics (BDA) is used to predict slice resource requirements and to estimate if a new slice request is accepted, then no service degradation will take place for incoming slice request and already working slices. In [62], cell load is predicted by forecasting UE mobility and traffic periodicity. The UE mobility is modeled using the self-similar least-action human walk (SLAW) and the probability for the UE to belong to a specific cell. Traffic is random, but is modeled to have periodic components. Slice requests are accepted within the predicted overall capacity for future time window. Performance analysis shows a more effective forecast if the observation window is larger; however, this requires more storage and computing resources. When the future time window is increased, the predictions become less accurate but require fewer resources. When a tenant's UEs are spread over fewer cells, the predictions are more accurate.

One solution is to consider provision queues if the resources for slices are not available now, but later. In [63] a queuing model is used for slice admission for impatient tenants. Multiple queues are modeled, with request balking (reluctant to join the queue) and reneging (leave queue after waiting). The results show that slice performance depends on the admission strategy.

Slice instantiation, that is, the creation of a slice instance, is covered in [64] and [65]. The 3GPP management-based architecture, called the E2E slicer [65] uses 3GPP defined CSMF, NSMF, and NSSMF. The authors evaluated the slice VNF instantation time in the case of uncached, cached, and container images of the VNF. The results showed that image caching and containers were the fastest to instantiate. Second, the UE attachment time was measured, and an additional message exchange with NSSF added 36 ms. This is acceptable performance. However, if the workload of the E2E Slicer reaches over 120 concurrent connections, the UE attachment time increases significantly, the NSSF workload needs to be monitored, and CPU resources should be added to save delay on UE attachment. In [64] a framework was proposed for automatic slice creation that enables the creation of more than one slice per service vertical. The test platform emulates LTE using OpenAirInterface with a FlexRAN controller. The results show that slice instantiation delay does not depend on how many slices are requested, the delay of UE handover to another slice does not depend on the number of slices exist and the UE attach time with dynamic slice instantiation is less than 3 s.

2.3.2 Slice modification (performance assurance)

The prepared RAN network slice subnets can overlap the existing resources many times. In the creation of the slice instance, it must be considered that resources are sufficient for all running slices. The performance of the infrastructure and performance of each slice must be monitored to gather information about resource usage and the fulfilment of requirements. In [66], the KPIs to monitor, how often to collect data, and the recommended thresholds for KPIs to detect the need for slice reconfiguration are presented. This threshold-based approach is also used by [67]. For slices that detected a low load (below the lower threshold) for a predefined time (more than one measuring cycle), a decision to downscale (scale-in) is made to decrease BS resources. For slices that detect a high load (above the upper threshold) for a predefined time, a decision to upscale (scale-out) is made to increase the BS resources. The parameter values for lower threshold, upper threshold and target load threshold for “breathe” of overloaded slice can be defined manually or in future authors proposed they can be learned automatically using decision trees in game theory.

Slice reconfiguration can be caused by a situation in which the infrastructure has to fit a new RAN slice; then, the existing RAN slices must provide some of their resources to a new RAN slice. As all slices must satisfy their SLA, the orchestrator selects slices that can provide resources to the new slice without violating their own SLA. Another cause of slice reconfiguration is the overload of one or more RAN slices. Slice overload can be detected by NSSMF, which can then trigger the overloaded slice to be scaled up and other slices to be scaled down. The other option of slice scaling is to handover the UEs in an overloaded slice into another not overloaded slice as proposed in [68] for V2X. The process of deciding what needs to be recond is complicated and time-consuming.

Slice provisioning includes slice admission with resource allocation. The slice descriptors in layers 1 (L1), 2 (L2), and 3 (L3) can be used to characterize the slice features, policies, and resources in RAN nodes [69], [54]. The L3 slice descriptor specifies the capacity allocation, radio resource management (RRM) policies, and configurations of radio resource control (RRC) protocol. The proposed configuration parameters are Slice Authorized Capacity and Slice Allocation Priority. For scheduling, the L2 descriptor defines L2 configuration parameters, such as slice scheduling priority and slice resource utilization (percentage of overall resources to be allocated to the slice). The L1 slice descriptor provides information regarding the radio resource structure of the cell.

A slice can be admitted to infrastructure (inter-slice admission) if resources are available. The slice admission decision depends on the resource availability. The resource requirement is evaluated from the SLA, and a new QoS flow for the tenant can be admitted if its SLA can be satisfied [70]. In [71] the UEs are admitted until sum-throughput with the best potential spectral efficiency found.

The optimization goal of resource allocation is to allocate the least resources as possible and use them as much as possible, with all slice SLAs satisfied. When there are more requests than existing resources can serve, then the acceptable requests must be chosen. The optimization goals can be spectral efficiency, maximized throughput, and SLA satisfaction. These objectives can be translated into profit (revenue) vs. loss, or reward vs.

penalty, and maximize profit or reward and minimize loss or penalty. In [70] the proposed admission control algorithm increased the overall throughput and profit by evaluating the penalty for not accepting flow and the extra revenue for accepting flow over slice SLA.

The amount of resources is fixed; however, the throughput that can be achieved depends on the radio signal quality, which can be considered stochastic in nature. The stochastic optimization is used to maximize slice throughput: in [71] UEs are admitted until sum-throughput with best potential spectral efficiency found, and in [72] the PRBs are allocated to the UE that is able to achieve maximum rate from this PRB at this slot.

In [73] the two-time-scale resource allocation is proposed. The inter-slice resource allocation includes resource reservation for groups of UEs (slices) with profit maximization, and the slot length is one hour. The short timescale slot length is 1–5 s, and contains intra-slice resource allocation under CSI uncertainty and traffic variation. The resource management lifecycle is one day to observe periodicity in UE traffic. To cope with stochastics in RAN resources, [74] propose resource overbooking for admission control and resource reservation to slices in a multi-operator core and shared RAN. Algorithms with resource overbooking with an optimization goal to minimize cost (penalty - reward) are proposed (one uses the Benders method; another heuristic algorithm uses knapsack problem solving). The simulation results show an increase of up to three times revenues increase compared to resource non-overbooking.

Multiple slice types are considered for the slice provision in [75]. The SLA mapping layer for joint slice admission and packet scheduler is proposed. The slices with strict requirements, the constant bit rate (CBR) and minimum bit rate (MBR) slices, obtain UEs admitted according to available resources, and the best-effort slice admits all UEs while the UE dropping rate is below the threshold specified by slice SLA. Resources are first allocated to the CBR slice, and the remaining resources are shared with MBR and BE slices with iterations until all MBR slices have their SLA met. The network resilience to slice traffic anomalies has been improved when centralized adaptation of slice traffic anomalies was used. Similarly, in [76] UEs are grouped into GBR, constant bit rate (CBR), minimum bit rate (MBR) UEs, and best effort (BE) UEs, and the UEs of each group are admitted with different criteria. The neural network is composed with inputs of request arrival rates and SINR parameters for each slice and slice control parameters. The output is KPIs for each slice: admission rate, dropping rate, average throughput are predicted. The UEs that request best-effort (BE) slices are all admitted to their slice, and if their active time is longer than the specified threshold, then the UE is dropped. For constant bit rate (CBR) slices, the UEs is admitted if the admission rate is above the threshold. For Minimum Bit Rate (MBR) slices, the KPIs should be below the average throughput and admission rate. A neural network consisting of three hidden layers with 50 nodes in each layer is used to predict the traffic.

For increasing the automation the slice life cycle is monitored continuously. For run-time slice LCM the continuous loop of pre-adaptation, pre-processing, multiplexing, allocation, and monitoring phases is proposed in [77]. This works on O-RAN's architecture.

2.3.3 Slice deletion

A slice will be deleted if its configuration is deleted from the slice database. The slice instance is deleted when it is decommissioned. The slice does not participate in admission, resource allocation, and scheduling, and these algorithms can work faster. Depending on the slice LCM KPIs proposed in [66], such as the slice deployment time and slice termination time, the decision to delete the slice when it is not needed is made.

2.3.4 Summary

The slice instantiation delay is achieved less than 1 min in [78] and approximately a minute in [64]. The slice handover delay is 2 s, which was achieved in [64]. The automation enables a significant decrease in the duration of slice LCM operation.

The management system manages the services and slices, and guarantees slice SLA satisfaction. The LCM determines what needs to be done if the SLA is not met. The RAN slice subnet is a logical network; thus, FCAPS management is applied. In the slice preparation phase, possible slices with RAN slice subnets are defined. At runtime, the slice instances must fit into the infrastructure and their SLAs are satisfied. When there are more slices than infrastructure is able to serve, then the decisions on what slice instances to run, are needed. The optimization goals are the efficiency of resource usage, provide maximum overall throughput, satisfy SLAs of all slices, and maximize revenue. The latter is a universal goal, as other goals can be transformed into monetary values. Performance is the key to trigger the slice modification LCM operation, and the slice configuration determines the NFs and resources allocated to the slice. Slicing enables dynamic modification of the network configuration to satisfy the slice SLA.

2.4 Subslicing

In the literature, the term subslice is often referred to as slice subnet in core, RAN, or other domains. I use the term subslice for a smaller part of the slice, which contains a subset of slice resources and subset of slice UEs. In 3GPP TS 23.501 [15] the slice identifier consists of two parts, slice/service type (SST) and slice descriptor (SD). The slices, with same SST and different SDs, can be considered as subslices, if they share the radio resources (band). In [79] the subslices are slices with same SST and different SD. The UE can connect to multiple slices. They evaluated the performance in core, and RAN was simulated. RAN had unlimited resources and physical layer not implemented, which means ideal radio quality and immediate transmission in the RAN is assumed.

The work in [80] proposed an algorithm to calculate how many UEs can be admitted to the subslice to achieve optimal subslice throughput. The number of UEs to admit is calculated based on optimal UE throughput, considering radio signal path loss, fading and shadowing. They report average cell throughput and signal-to-noise ratio many times higher than baseline results. However, the input UEs had variable requested rates, such that the requested sum rate for slice can be in the range of below and above theoretical capacity of 80 MHz BWP.

In [81] the UE is connected to multiple subslices. The subslices are a logical grouping of services with similar SLA values (throughput, delay, etc). The purpose is to maximize energy efficiency per UE per service. Their optimization results show that on average, 4 or 5 services are in one subslice. However, similar to previous work of the same authors, the requested sum rate of a slice is variable between capacity underload and overload. The usage of MIMO is not known, but still the achieved cell throughputs seem approximately, 15000 times higher than theoretical maximum throughput 1.33 Mbps per one RB (15 kHz subcarrier spacing).

In [82] authors propose to optimize services into subslices with maximum capacity and throughput. Two slices considered, which contain up to 40 and 91 subslices, respectively. The goal is the optimal service allocation for UE to achieve maximum capacity of the slice. System throughput was at least 5 times higher than compared resource allocation methods.

In [83], networks (packet core, core cloud, edge cloud, RAN) are divided into smaller NSSIs (network slice subnets called subslices) and each UE selects many suitable subslices to use. UE creates a slice which contains all selected subslices. If UE uses multiple applications, it can attach to e.g. multiple core and edge subslices. This architecture enables UE to be connected to multiple slices simultaneously, which has been an open issue in research. They proposed a subslicing method where a subslice in a RAN is treated as a virtual cell that includes multiple physical cells. This approach aims to improve the slice performance by reducing the signalling required for cell handovers within the RAN subslice. Simulations are done with RAN subslices, called virtual cells, which are in radius of number of physical cells. Performance improved because signalling required to perform physical cell handover, caused by UE mobility, has reduced. However, the performance improvement is indirect, the latency and throughput can be improved because UE handover is done faster.

The work presented in [84] proposed a promising subslicing method. First, the UE features were selected using a support vector machine (SVM). Secondly, the UEs were clustered by these selected features using k-means into several numbers of clusters. The quality of cluster was evaluated by using the Silhouette coefficient, and the best number of clusters was selected as the number of subslices. However, this approach did not consider the performance when creating the subslices, which can lead to a poor performance for small subslices. Their simulations were conducted using Android UEs in Wi-Fi, and the performance of 5G-NR RAT was not evaluated. The performance improvement is reported in decrease in bandwidth consumption, improved load balancing, latency, throughput and energy efficiency.

The bandwidth subpartitioning has some works which are related to subslice bandwidth allocation. The flexible placement of allocated radio resources on resource grid is considered in [85], [86] and for multicell in [87]. The objective is to efficient bandwidth use and move used BWPs close to each other to save bandwidth. However, the BWP size from the slice performance perspective is not considered.

2.4.1 Summary

Network slice subnets are often called as subslices. Sometimes slices with same SST but different SD are called subslices. RAN subslicing in the context of this thesis is a slice bandwidth subpartitioning and slice UE grouping. One research article has done similar subslicing. They focused to group UEs by their requirements and to efficient allocation of multiple resource types in RAN.

The RAN slice performance dependence on RAN slice BWP size is not researched in context of subslicing for slice performance improvement.

2.5 5G RAN NR Emulators and Simulators

2.5.1 Emulators

The emulator duplicates 5GS software and hardware features. The hardware is emulated by software defined radio (SDR) hardware and software is 5G RAN emulator. For research, the small-scale emulated RAN is needed to study gNB-UE links and RAN system with slices. This can be implemented with SDR that must be compatible to 5G standards to enable emulated one or multiple UEs or Commercial off-the-shelf (COTS) UEs to connect to emulated gNB-RU or multiple gNB-RUs. The gNB nodes should support controlling by RAN controller such as RIC and provide data used to FCAPS management. The overview of

Table 3: 5G RAN emulators

Name	Paper, year	Features	Supported SDR hardware	RAN slicing support	License	Open source	Supported OS
O-RAN software for RAN		5GS, network slicing in CN and RAN, cloud, gNB virtualization and splitting into CU, DU and RU	Commercial O-RU hardware	Yes	Apache 2.0	Yes	Linux
OpenAirInterface	[88], 2020	NSA and SA gNB, eNB, 4G and 5G UEs, 4G core, 5G core, COTS UE connection	Ettus USRP B210, X310 and N310	With RAN controller	OAI Public License V1.1	Yes	Linux
srsRAN	[89], 2016	4G core, 5G core, eNB, gNB, 4G UE, 5G UE, COTS UE connection	Ettus USRP B2x0/X3x0 families, BladeRF, LimeSDR	No	AGPLv3 and commercial	Yes	Linux
free5GRAN	[90], 2021	5G UE, cell search, band scanning	Ettus USRP B210, N210 and X310	No	Apache 2.0	Yes	Linux

5GS emulators and software is given in [91]. The information about mature 5G emulators such as O-RAN software for RAN [92], OpenAirInterface [93] and srsRAN [94] and one new emulator free5GRAN is included in Tab. 3.

O-RAN Software community has full stack of software [92] to emulate/deploy 5GS. Current latest release is G release (December 2022) has basic RAN slicing support. The gNB has been split into three parts, O-CU as central unit, O-DU as distributed unit and O-RU as radio unit. The control plane and user plane are separated in central unit as O-CU-CP communicates in CN with AMF and O-CU-UP communicates in CN with UPF. The SMO and near-real-time RIC control all RAN entities via interfaces O1 and E2 respectively.

The OpenAirInterface (OAI) [93], presented in [88] is an open source project to implement LTE and 5G NR on general purpose computers and SDR hardware. The gNB, a 5G base station, is implemented for EN-DC (E-UTRA-NR Dual Connectivity) non-standalone (NSA) mode. The base station is split into CU (implements RRC and PDCP layers), DU (implements physical, MAC and RLC layers and RU (antenna). The interfaces between CU and DU is F1 defined by 3GPP, inside DU the interface between physical and MAC layer is Functional Application Platform Interface (FAPI). The DU is split into DU and RU by O-RAN 7.2 split option. The SDR hardware used are Universal Software Radio Peripheral (USRP) B210, X310 and N310. The gNB supports 3GPP Rel-15 physical layer, uses static scheduler with a single preconfigured UE and statically allocated resources. A "noS1" mode to work without the support of core network is implemented. In the core 3 main NFs are in development - AMF, SMF and UPF and interfaces to support minimal functionality of 5G. COTS UE Oppo Reno 5G phone was used successfully to connect gNB. 5G core implementation is work in progress [95].

Although srsLTE, presented in [89], implements LTE, it is widely used to evaluate proposals for 5G. The srsLTE, is an open source LTE-compliant platform for SDR to experiment with LTE or LTE and Wi-Fi coexistence in unlicensed bands. In 2022 srsLTE has been renamed to srsRAN and it has LTE Rel-10 compatible UE with features up to Rel-15 and eNB with Rel-10 aligned with 5G non-standalone support. The lightweight implementation of LTE core, srsEPC is available. So srsRAN is now free open source full suite of LTE and 5G network [94]. The srsRAN supports carrier aggregation, multicast/broadcast traffic, NB-IoT, can run on Raspberry Pi, can receive and decode c-V2X signals, connect COTS UE to eNB and enables to simulate IQ samples between UE and eNB without physical SDR hardware.

For 5GC implementation a free5GC [96], one lightweight free open source software has been used. The 5G RAN emulator named free5GRAN, presented in [90], is developed for studying 5G RAN. It currently has UE implemented to be connected with SDR hardware and is able to detect 5G NR cell and decode some signals.

For a conclusion, there exist strong 5GS emulation platforms such as O-RAN software, OpenAirInterface and srsRAN. The latter two do not support RAN slicing, but necessary MANO entities and RAN controllers can be developed on top of those emulators.

2.5.2 Simulators

The simulator mimics the behaviour and configuration of real 5GS. The simulators are mainly needed for radio network planning to estimate number, location and configuration of RAN nodes [97]. The coverage, capacity and cost are evaluated.

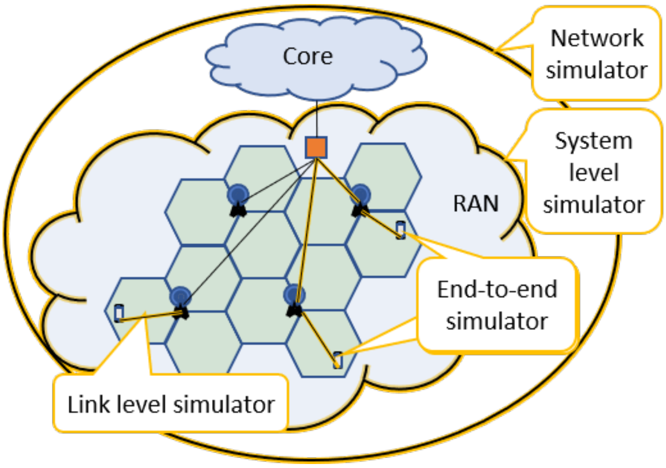


Figure 7: Types of simulators

The different types of simulators are shown in Fig. 7. The simulator types are system level simulator to evaluate networking issues and link level simulator to evaluate the link issues. Hybrid simulator combines both link and system level simulators. End-to-end (E2E) simulator simulates the communication channel from transmitter to receiver. Network simulator mimics the whole network.

The simulator suitable for evaluation of RAN slicing solutions should not just be capable to simulate 5G with its variety of verticals and use cases, but also be capable to simulate simultaneously working multiple logical networks on top of simulated physical infrastructure for MANO of the both slices and infrastructure. The relevant KPIs for RAN slicing are performance KPIs. The simulator that enables to simulate RAN slicing in 5G should be able to create delays and additional control data stream related to RAN slicing. In addition to performance of each slice, the overall performance data is needed.

Simulators can be simpler than emulators. The packet level simulation is too slow to simulate in real time for example the mMTC traffic with short packets and long transmission cycle. Often the whole core is not needed when only RAN is in interest. These simulations can be replaced statistical models of mMTC traffic and statistical model of delay caused by core NFs [98].

There exist one slicing simulator and several 5G simulators that are promising to enable RAN slicing in future. Comparison of simulators is in Tab. 4.

The network slicing simulator Slicesim [106] enables to define slices with slice settings, BSs and amount of resources for each slice, UE settings and UE mobility patterns. The output is a graph with overall network statistics. This is a simple RAN slicing simulator, that enables to simulate slices of RAN resources present on RAN BSs. The output data gives overall statistics, but not per slice statistics.

The well-known network simulators ns-3 and OMNeT++ have extensions and libraries that enable to simulate 5G. The 5G LENA [107] is a ns-3 extension to support 5G NR [103] and NFV and SDR on physical layer [108]. The support of RAN slicing is currently implemented by BWP management. The other ns-3 based simulator, 5G K-Simulator [109], presented in [102] includes SDN architecture for virtualization and mmWave module. RAN slicing is not supported; however, the virtualization makes it promising. The simulation of disaggregated gNB placed on multiple PoP DCs enables to evaluate utilization of computing, storage and networking resources of gNB, slices and overall resources.

Simu5G, introduced in [105] is an OMNET++ library to simulate 5G gNB and 5G UE, however no slicing simulation is implemented. But its ability to support real-time emulation makes it useful to be part of the emulation system of network slicing in 5G.

The system level simulator Wireless Simulator Evolution (WiSE) for 5G mobile networks is proposed in [101]. It can simulate 5G verticals such as eMBB, URLLC, mMTC. If multiple simulators are running simultaneously and a framework on top of it collects statistics, then a slicing simulator can be composed. Another system level simulator called 5G-air-simulator [110] proposed in [104] is useful to watch as RAN slicing is planned for future work.

The Matlab-based Vienna 5G Simulators contain system level [99] and link level [100] simulators enable to simulate multiple BSs, UEs with movement and evaluate performance of large networks. However, the slicing in RAN is not supported. The Matlab has a 5G Toolbox [16] with link level, system level and E2E simulators but no RAN slicing implemented.

For RAN slicing support, simulators need to implement a gNB with dynamic resource partitioning to slices and UE association with slice and base station. In addition to overall performance data, the slice and UE performance data should be available as output. If more resources and time available one should consider emulating the 5G RAN with slicing enabled as most of the gNB and UE consists of software and by modification of software the new features can be implemented and evaluated.

Table 4: Simulators

Name	**Paper, year**	**Type of simulator**	**Features**	**Slicing support**	**Licence**	**Open source**	**Supported OS**
Vienna 5G Simulators	[99], [100], 2018	System level and link level	Large scale networks with multiple types of BSs and moving UEs for performance evaluation	No	Proprietary, free for academic use	Yes	Windows, Linux, MacOS
WiSE	[101], 2018	System level	Multiple BSs and UEs, scenarios, ITU-R channel models	Supports eMBB, URLLC and mMTC	Proprietary	No	Windows, Linux
5G K-simulator	[102], 2018	Link level and system level	Network simulator based on ns-3; BSs, UEs; PHY, MAC, RLC, PDCP, RRC;	No	Free, no license	Yes	Linux
Slicesim	2019	Discrete event	Slices, UEs, BSs	Yes	MIT license	Yes	Windows, Linux, MacOS
5G LENA	[103], 2019	System level and link level	5G NR extension to ns-3 LENA; gNBs, UEs; schedulers, BWP manager, multiple numerologies; PHY, MAC, RLC, PDCP.	Implemented by BWP management	GPLv2 license	Yes	Linux
5G-air-simulator	[104], 2020	Event driven system level	PHY, MAC, RLC, PDCP; cells, sectors, UE mobility, path loss models, MIMO, NB-IoT	Planned as future work	GNU GPL	Yes	Linux
Simu5G	[105], 2020	System level and link level; discrete event	OMNeT++-based model library for 5G; gNBs, UEs; PHY, MAC, RLC, PDCP; channel model, carrier aggregation; UPF, EN-DC; D2D, MEC; real-time emulation	No	LGPL	Yes	Windows, Linux, MacOS
Matlab 5G Toolbox		System level, link level and E2E	5G NR simulator of PHY layer signals	No	Proprietary	Yes	Windows, Linux, MacOS

2.5.3 Summary

In implementations of the O-RAN-compatible architecture, it is preferred and developed to be interoperable with 3GPP/ETSI management system architecture. O-RAN has standardized open interfaces as this enables to deploy architecture with multivendor components.

The OSM concentrates more on service and VNF management, and TN is left out of scope. The SMO, ONAP and Aether all use architecture defined by O-RAN. ONAP has developed explicit TN that connects all nodes of disaggregated RAN, into TN slice subnet. The ONAP is the most flexible to enable each of 3GPP's management function to work as native or external with its implementation.

The RAN controller is the node that directly controls the RAN nodes to configure RAN infrastructure to enable E2E network slicing in RAN domain. RAN controllers have built-in intelligence that can be implemented with microservices and applications. To implement network slicing in RAN, the two-level schedulers are controlled by RIC. All RICs fit into O-RANs architecture, with open interfaces A1 on NBI and E2 on SBI.

All RAN emulators discussed support SDR hardware and are able to work without SDR hardware to simulate physical layer of 5G NR and LTE. Both RAN simulators and emulators are needed for evaluation of different system components at the other end of the interface to RAN.

2.6 Conclusion

In this chapter, an overview of the network and RAN slicing architecture in 5G was presented. In order to ensure efficient resource utilization while maintaining QoS requirements, resource allocation plays a critical role.

The resource allocation and scheduling for RAN slicing has been extensively researched. The efficiency of resource utilization, bandwidth allocation, and energy efficiency from one side and the SLA satisfaction on the other side can be achieved by optimized resource allocation. This leads to the close-to-slice-overload situation. The slice resources can be adjusted by slice LCM while the slice instance is running and during the modification phase of the slice life cycle.

Radio resources are allocated by using RAN schedulers. New schedulers can satisfy both throughput and delay requirements for same and different slices/UEs. The slice BWP size depends on the amount of radio resources that required to be allocated. The slice performance dependence on the slice BWP size is not investigated, moreover it is not considered affecting the slice performance.

The service categorization into verticals enables to allocate different resources to different types of services. Similarly, the subslicing tries to group UEs into subslices by their SLA QoS requirements. However, it has not been investigated whether other UE features could be used for grouping UEs into subslices.

There are several 5G NR simulators and emulators available, however only a few support RAN slicing.

3 Evaluation of the RAN Slice Performance

In this chapter, the performance dependence on a slice or subslice size is investigated. In the context, the slice can contain subslices, but if the slice does not contain subslices, the performance dependence on slice size is the same as for subslice. In my publication 1 [1] it was discovered that slice performance depends on the number of subslices within a slice. The subslices are smaller if the slice is split into more subslices. This leads me to further investigate the subslice performance dependence on the subslice size.

Subslice performance was evaluated for all possible subslice sizes, and two datasets were collected. Dataset 1 was used to propose methods to improve slice performance by subslicing in my publications 2 [2] and 3 [3], and dataset 2 was used to find the best subslice sizes and training data labeling for subslice decisions in my publication 4 [4]. If the performance of 5G-NR network depends on BWP size, then we can use subslices of appropriate size to improve RAN slice performance on limited slice BWP. These results are the base for development of methods proposed for subslicing in next chapters.

3.1 Simulation Planning

The goal of bandwidth allocation optimization is resource usage efficiency and satisfaction of SLAs. Each RB allocated to a subslice must be consumed by a UE to evaluate the performance of a subslice of any size.

3.1.1 Simulation Tool

The tool to simulate subslices is MATLAB 5G Toolbox system-level simulation tool called NR Cell Performance Evaluation with Physical Layer Integration [16]. The performance dependence on BWP size lies on the physical layer. Channel quality indicator (CQI) is measured using demodulation reference signals (DM-RS) and channel state information reference signals (CSI-RS). Based on CQI, the modulation and coding scheme (MCS) is selected from table 5.2.2.1-3 from 3GPP TS 38.214 [111]. From MCS, the transport block size (TBS) is calculated as in Section 5.1.3.2 (DL) and 6.1.4.2 (UL) of 3GPP TS 38.214 [111].

The user needs to specify the BWP and UEs. The results are collected from simulation logs and metrics.

3.1.2 Simulation Time

The radio resources do not have 1:1 mapping to achieved rate, because radio signal quality has stochastic nature. In [66] it is recommended to collect performance data for 30 s as base for evaluation.

The simulation time of 30 s for a slice of maximum size serving a UE per RB takes up to one week on computers with specification of i5-4570, CPU @ 3.20 GHz, 8 GB RAM. To simulate one second of a slice working, the simulation takes 12–14 h.

The simulation time 1 s (100 frames) is selected as a tradeoff between time it takes to simulate and quality of the achieved results.

3.1.3 Subslice Sizes

All possible subslice sizes in granularity of one RB in frequency scale are simulated.

The minimum 5G-NR BWP size requirement is not specified in 3GPP standards. In [112] it is claimed that if the BWP is smaller than 20 physical RBs, then the different control blocks can puncture each other and increase the coupling loss in 5G NR. This degrades the performance for UEs. From this, it can be concluded that larger BWPs are better. But for the UE, the smaller bandwidth to use, the more energy efficient it is [113].

The maximum subslice size is 275 RBs. This is 50 MHz, if subcarrier spacing is 15 kHz [9]. If a specific BWP is used, then its maximum size is the maximum subslice size.

In MATLAB, it is possible to simulate subslices in size of 4-275 RBs. For each RB allocated, the UE is admitted which is able to consume the RB by its requested rate.

3.1.4 Carrier Frequency

For dataset 1 the carrier frequency is 3 GHz, an arbitrary frequency in FR1.

For dataset 2 the real band from FR1 is used, n28 (FDD, UL 730-748 MHz, DL 758-803 MHz) [9], subcarrier spacing is 15 kHz thus maximum subslice bandwidth is 250 RBs. The band n28 can be used for both machine to machine (M2M) IoT and vehicle to everything (V2X) service types.

3.1.5 UE Requested Rates

To calculate the requested rate for UE to be able to consume the RB, the well-known Shannon's formula [56] can be used.

$$C = B \cdot \rho, \tag{1}$$

where C is the channel capacity in bps, B is bandwidth in Hz and ρ is spectral efficiency in bps/Hz.

In 3GPP TS 38.214 [111] the spectral efficiencies for 5G-NR are shown. The theoretical maximum spectral efficiency for DL is 7.4063 bps/Hz as shown in table 5.1.3.1-2 and for UL is 5.5547 bps/Hz as shown in table 6.1.4.1-1. Considering the subcarrier spacing 15 kHz, the maximum theoretical capacity of one RB, with bandwidth 180 kHz, is 1.33 Mbps for DL and 1.00 Mbps for UL.

Simulation results in my publication 1 [1] show that if each UE requests the 50% of theoretical maximum rate of RB, then the allocated RBs are consumed in close to slice overload at maximum slice size without subslicing.

3.1.6 UE Packet Sizes

Typical packet size examples for verticals are 1500 Bytes for eMBB, 160 Bytes for URLLC and 40 Bytes for mMTC vertical [24], [25].

Packet sizes of 1500, 500, 15, 50 and 40 Bytes were selected for simulations.

3.1.7 UE BLER

The UE position can be set in 3-D space, where gNB is located in coordinates (0,0,0). By setting the values of x-coordinate, the distance from gNB can be set.

The distance has the effect on signal quality, which has the effect on BLER. Thus, by setting UE's position, I can affect its achieved BLER in the simulations. However, the MCS is selected to target the UE BLER below 0.1. On the other hand, the UE distance alone does not determine BLER, but used BWP size has its effect on UE BLER as well in addition to channel characteristics.

3.1.8 Other Settings

Other simulation settings are shown in Tab. 5. I will not evaluate the scheduler and select the common scheduler Round Robin. This scheduler suits for a group of UEs with similar requested rates and BLER [41].

The simulation is run once because I do not prove that the 5G-NR performance is similar to this but show what can be done if the 5G-NR performance is similar to this. The proposed methods and algorithms presented in this thesis are based on these simulation results of slice performance dependence on the slice BWP size. If the actual RAN slice has a different slice performance dependence on slice BWP size, the new training dataset must be collected and labeled for the NN described in Section 5.

Table 5: MATLAB Toolbox settings

Parameter	**Value**
Channel model (for both UL and DL)	CDL-C
PUSCH preparation time for UEs	200 μs
Logical channels per UE	1
RLC entity type	UM bidirectional
Duplex mode	FDD
Scheduler strategy	Round Robin
Length of scheduling cycle	1 frame
RB allocation limit UL	same as RBs for subslice
RB allocation limit DL	same as RBs for subslice
Simulation time	1 s
Subslice simulation tool from MATLAB 5G Toolbox	NR Cell Performance Evaluation with Physical Layer Integration [16] R2021b

3.1.9 KPIs to be Collected

For performance evaluation, the utilization of any resource is the KPI [66]. I collect values of subslice RB utilization in UL (utilUL) and sublsice RB utilization in DL (utilDL). The used RBs are counted and utilization is calculated as the ratio of used RBs to allocated RBs.

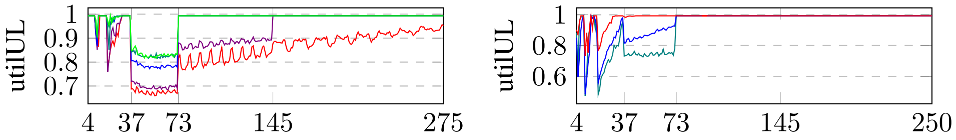
The throughput is one KPI, often used. However, I collect goodput, that is application-level throughput, to evaluate the transmitted useful data without packet headers and retransmissions. In MATLAB R2021b implementation, the throughput contains all retransmissions, while goodput does not. Packet headers (RLC header is 4 bytes) are counted in both throughput and goodput. To make the goodput results comparable for all subslice sizes, the goodput per one RB is calculated by dividing subslice goodput to number of subslice RBs. Goodput KPIs are goodput per one RB in UL (gdp1UL) and goodput per one RB in DL (gdp1DL).

The subslice BLER is the average of all UE BLERs in UL (blerUL) and in DL (blerDL), respectively.

In addition, the spectral efficiency was measured, but it follows the pattern of goodput per one RB and thus not collected. The buffer size at the end of simulation can measure the delay indirectly. Delay is another relevant KPI along with throughput/goodput; however, more engineering work is required to obtain the latency or delay values from the MATLAB 5G-NR simulator, and it was not collected.

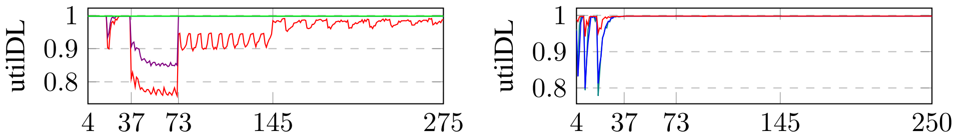
3.2 Results

I present two datasets which are used as subslice performance dependence on subslice size to propose methods for decisions in MCCL in Chapter 5. The results are shown in Fig. 8; left column contains dataset 1, right column contains dataset 2, both are described in following subsections.



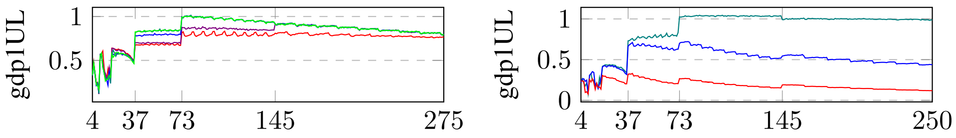
(a) Bandwidth utilization in uplink (utilUL)

(b) Bandwidth utilization in uplink (utilUL)



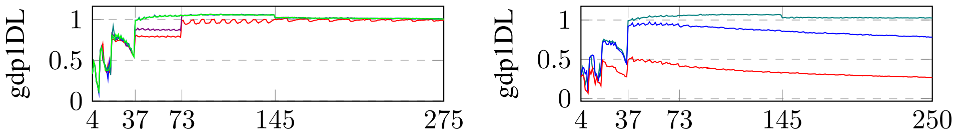
(c) Bandwidth utilization in downlink (utilDL)

(d) Bandwidth utilization in downlink (utilDL)



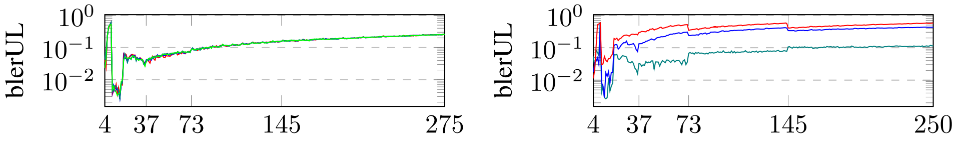
(e) Goodput per one RB in uplink (gdp1UL)

(f) Goodput per one RB in uplink (gdp1UL)



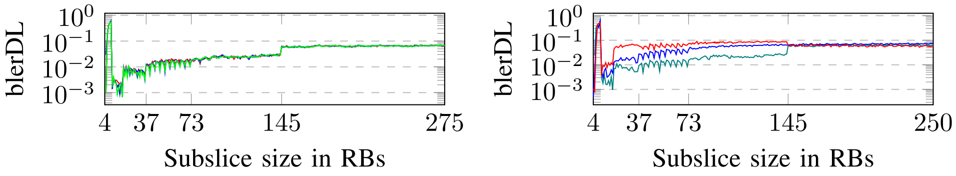
(g) Goodput per one RB in downlink (gdp1DL)

(h) Goodput per one RB in downlink (gdp1DL)



(i) Slice BLER in uplink (blerUL)

(j) Slice BLER in uplink (blerUL)



(k) Slice BLER in downlink (blerDL)

(l) Slice BLER in downlink (blerDL)

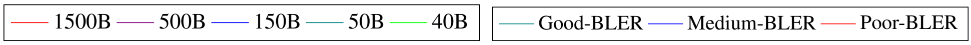


Figure 8: Subslice size simulation results. Left column contains dataset 1 from my publication 3 [3], right column contains dataset 2 from my publication 4 [4].

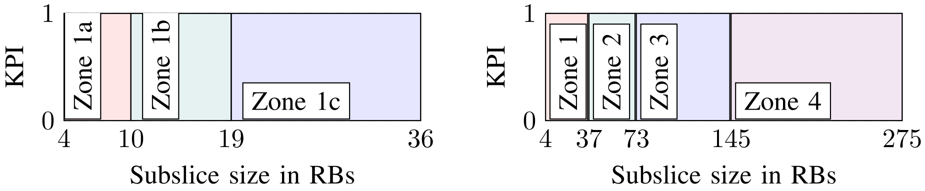
3.2.1 Dataset 1

This dataset has carrier frequency for BWP start selected arbitrary, 3 GHz. The maximum subslice size is 275 RBs. UEs are expected to achieve good BLER with the distance from gNB is set in range of 0 to 174 m according to simulation trials. The requested rate for each UE is 500 kbps in UL and 667 kbps in DL. Dataset contains 5 sub-datasets where the requested rate is served using packets in sizes of 1500, 500, 150, 50 and 40 Bytes.

3.2.2 Dataset 2

This dataset uses 3GPP band n28, its maximum size is 45 MHz (250 RBs, subcarrier spacing is 15 kHz). This can be used for the mMTC or V2X vertical. Each UE requests 200 kbps in both UL and DL using packet size of 40 Bytes. Dataset contains 3 sub-datasets where the UE BLER is expected to be achieved good, medium and poor by setting UE distance from gNB in range of 1-250 m, 1000-1250m and 6000-6250 m, respectively.

3.2.3 Findings



(a) Performance zones in zone 1

(b) Performance zones

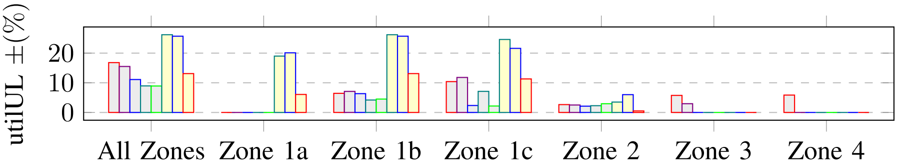
Figure 9: Performance zones based on slice size and performance.

From the results shown in Fig. 8 it can be seen several performance zones. Small subslices, in size of 4-36 RBs are in Zone 1. Within this zone there are three distinct performance zones, where the BLER has great differences: in Zone 1a the BLER is high, in Zone 1b, the BLER is low, and the BLER has intermediate values in Zone 1c, see Fig. 9a. The goodput is low in both zones 1a and 1b, but it starts to increase in zone 1c. Utilization is high, with low peaks on zone boundaries. Other performance zones are shown in Fig. 9b. The Zone 4 is a high goodput zone, but only if UE BLER is good. The Zone 3 is a zone of high goodput, but BLER is lower. The Zone 2 is the best utilization zone with a reasonable goodput.

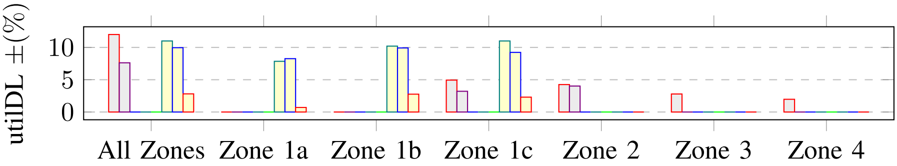
The more detailed overview of the ranges in KPI values in percentages is given by Fig. 10. By selection of the size of the subslice the utilization in UL can be changed up to $\pm 26\%$ and utilization in DL up to $\pm 12\%$. The utilization does not change much for larger subslices because it is close to slice overload, as was evaluated at the selection of UE requested rates (see Fig. 10a and 10b). The goodput per one RB in the subslice can be changed by up to $\pm 46\%$. Similar to utilization, the goodput changes less in Zones 2, 3 and 4, except if UE BLER is other than good (see Fig. 10c and 10d). The average BLER of subslice UEs can be changed by up to $\pm 50\%$ by selection of the subslice size. The highest BLER is in Zone 1a, and the lowest BLER is in Zone 1b. The BLER changes the least amount in Zone 4 (see Fig. 10e and 10f).

The dataset 1 performance depends on used packet size as follows. With smaller packets, both utilization and goodput are higher. BLER does not depend on packet size. The dataset 2 performance depends on UE BLER as follows: If UE BLER is higher, then utilization is higher and goodput is lower.

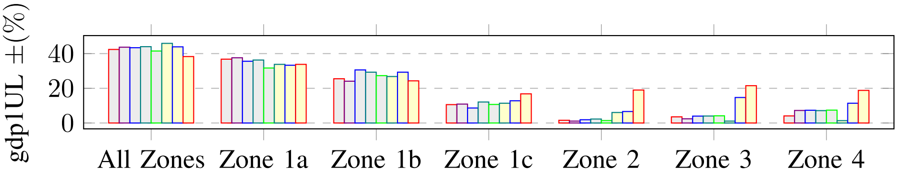
I investigated the cause of poor performance of small slices in both 3GPP TS 38.214 [111] and MATLAB implementation of 5G-NR. Smaller BWP has less reference symbols to use for channel estimation and channel is not estimated correctly to select proper TBS. If channel is better than estimated, then too small TBS is selected and achieved rate is low. If the channel is worse than estimated, then block error occurs. The performance pattern dependent on BWP size can be repeated if average MCS of UEs is calculated for each BWP size and excluding MCSs for first 3 slots when CSI is not available.



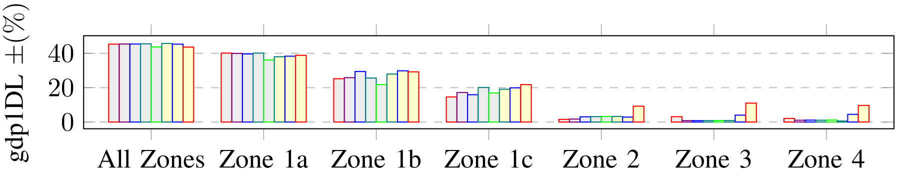
(a) Bandwidth utilization in uplink (utilUL)



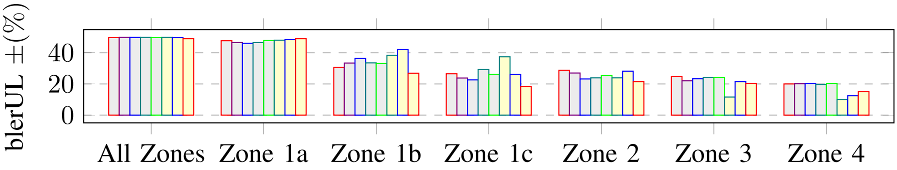
(b) Bandwidth utilization in downlink (utilDL)



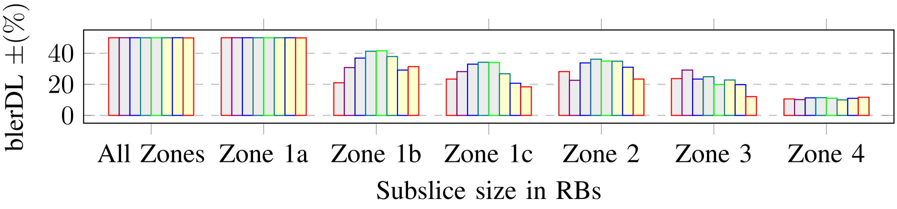
(c) Goodput per one RB in uplink (gdp1UL)



(d) Goodput per one RB in downlink (gdp1DL)



(e) Slice BLER in uplink (blerUL)



(f) Slice BLER in downlink (blerDL)

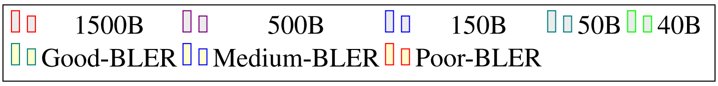


Figure 10: Ranges of KPI values in percentages in different performance zones of subslices in all possible sizes.

3.3 Conclusion

In this chapter, the performance dependence on slice or subslice size was investigated. The subslices of all possible sizes were simulated using MATLAB 5G toolbox tool. For each RB allocated, the UE is admitted which is able to consume the RB by its requested rate. The utilization, goodput per one allocated RB and BLER were evaluated for both UL and DL. By selecting the subslice size, the utilization can be changed by up to 26% in UL and 12% in DL, the goodput per one RB can be changed by up to 46% in both UL and DL, and the BLER can be changed by up to 50% in both UL and DL. The subslice performance dependence on subslice size is documented in this chapter for next contributions.

4 RAN Subslicing Algorithms

To perform subslicing I need to group UEs into smaller groups and divide slice bandwidth between these smaller UE groups. In the state of the art it is not known how the UE grouping into subslices affects slice performance. The subslicing does not create any new RBs, thus the overall RBs should be divided to UE groups such that groups which need more RBs will get more, but groups which have sufficient RBs, will not lose significant number of RBs. Otherwise, the UEs with poor performance will improve their performance in subslices at the cost of UEs with good performance. In any case, this cost should not be too high.

In the previous chapter, it was discovered that small subslices exhibit poor performance. In this chapter, the proposed subslicing algorithms should prevent to create too small subslices. The subslicing algorithm called user clustering with bandwidth allocation (UCwBA) is based on dataset 1 and published in my publication 2 [2]. The subslicing algorithms called dynamic subslicing by rate (DSbR) and dynamic subslicing by BLER (DSbB) are based on dataset 2 and published in my publication 4 [4].

4.1 Minimum Subslice Size Requirement

Clustering is a grouping of a set of objects by their similarity.

The minimum subslice size requirement is the minimum subslice size in RBs which can achieve a reasonably good performance. For UE clustering, it needs to be converted to minimum cluster size.

If all UEs request the same rate, then the minimum subslice size, S_{min} can be converted to minimum cluster size, C_{min} using the equation:

$$C_{min} = \left\lceil S_{min} \cdot \frac{N^{(UE)}}{N^{(RB)}} \right\rceil, \tag{2}$$

where $N^{(UE)}$ is the number of slice UEs and $N^{(RB)}$ is the number of slice RBs.

If UEs in the slice request different rates, the requested sum rate is used. The slice requested sum rate is a sum of requested rates of the slice UEs, and it is used to calculate the cluster minimum requested sum rate, R_{min} as follows:

$$R_{min} = \left\lfloor S_{min} \cdot \frac{R^{(s)}}{N^{(RB)}} \right\rfloor, \tag{3}$$

where $R^{(s)}$ is the slice sum rate. When the sum rate of UEs in the cluster is greater than the minimum requested sum rate, then this cluster has sufficient number of UEs.

In subslicing algorithm, if RBs are allocated proportional to the cluster BLER, then $N^{(RB)}$ is the number of slice RBs, which are allocated initially to the subslices either proportional to the number of UEs or proportional to the sum rate.

4.2 UE Clustering

The purpose of UE clustering is to have multiple smaller groups of slice UEs. The UEs in one group do not need to be similar to each other (if it is not possible), but the size of the UE group must be sufficient to allocate the number of RBs as subslice minimum size constraint.

The purpose of the clustering algorithm is to group similar objects into the same cluster. For subslicing, the first objective is to divide the slice UEs into groups with sizes not less than a size constraint and the second objective is to group similar UEs in the same group as possible.

The k-means [114] is a well-known clustering algorithm based on centroids. Centroid is in the center of the cluster and its location is calculated as the mean of all data points in the corresponding cluster. Initially, the locations of centroids are random. The k-means algorithm consists of three steps: distance calculation, cluster assignment, and a new centroid calculation. The algorithm finishes at convergence when the centroids do not move. Cluster size means that how many data points are close to the centroid of the cluster. The cluster size depends on the number of data points that are closer to the one centroid and not to the other centroids. The cluster size is not limited. It can be empty or contain all data points. Furthermore, there are no outliers as each data point is assigned to a cluster.

K-means has many modifications, ranging from unsupervised (no number of clusters given [115]) to balanced (all clusters of the same size [116]). The constrained k-means [117] algorithm enables the definition of constraints such as the minimum cluster size, and the optimization problem is solved using a linear programming method.

I propose modified k-means in my publication 2 [2], which creates the required number of clusters which size is at least the specified minimum cluster size. Contrary to original k-means, where a data point selects a cluster whose centroid is the closest, in the proposed modification, the cluster selects the data point which is the closest to its centroid. This modification works until all clusters have their minimum size requirement satisfied. The rest of data points are clustered using original k-means. The Fig. 11 illustrates both original and modified k-means algorithms.

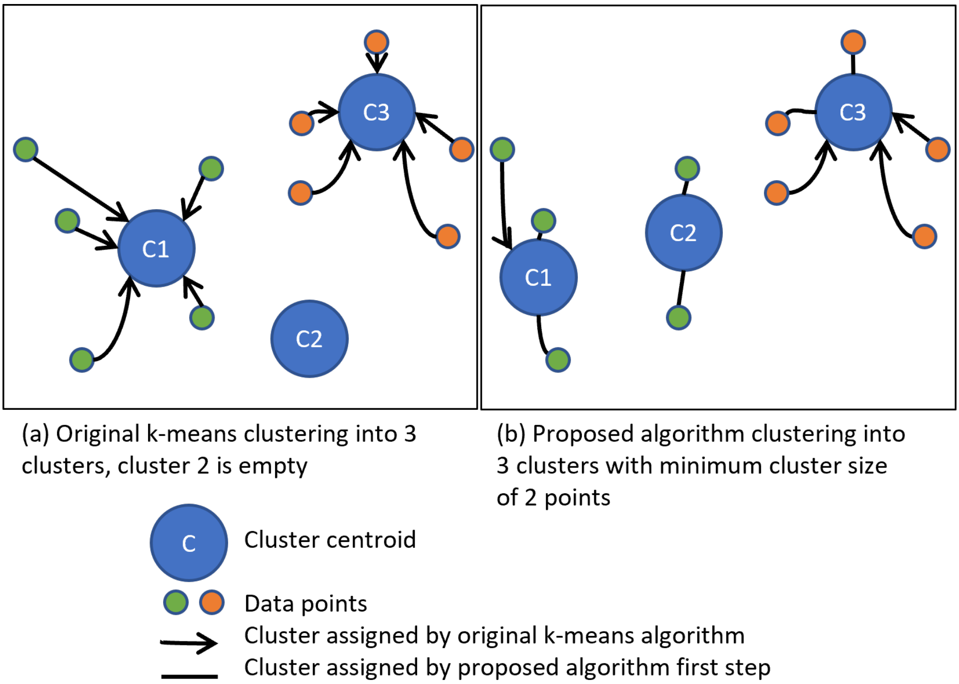
4.3 Bandwidth Allocation

The scheduler allocates RBs to the UEs. The bandwidth allocation for subslices allocates RBs to the groups of UEs inside the slice. If subslice contains UEs with poor BLER, it needs more RBs for retransmission. From my simulation results in my publication 1 [1], it can be seen that although the requested sum rate is the same, if more UEs are in the slice then the slice utilization is higher compared to if less UEs are in the slice. The RBs needed by certain subslices are reallocated from other subslices. The smaller the slice, the more sensitive it is to RB reallocation. The adjustments can be really small, because one RB is 180 kHz in bandwidth.

The slice RBs are allocated in 3 steps (see Fig. 12):

1. initial allocation,
2. second allocation,
3. third allocation.

Initial RB allocation is different in subslicing algorithms UCwBA from my publication 2 [2] and subslicing algorithms DSbR and DSbB from my publication 4 [4]. In subslicing algorithm UCwBA initially RBs are allocated proportionally to number of UEs because all UEs requested same rates. In subslicing algorithms DSbR and DSbB initially RBs are allocated proportional to subslice requested sum rate. About 95-99% of RBs are allocated initially. Second RB allocation is proportional to subslice BLER. The third allocation is for leftover RBs which are allocated to the subslice where is the UE with the highest BLER. Details about proposed subslicing algorithms can be found in my publications 2 [2] and 4 [4].


C3
C1
C2
C3
C2
C1
C
Cluster centroid
Data points
Cluster assigned by original k-means algorithm
Cluster assigned by proposed algorithm first step

(a) Original k-means clustering into 3 clusters, cluster 2 is empty

(b) Proposed algorithm clustering into 3 clusters with minimum cluster size of 2 points

Figure 11: An example of how the proposed clustering algorithm clusters data points into three clusters with a minimum cluster size of 2. The data points cluster well into two clusters (green and orange). Figure from my publication 2 [2].

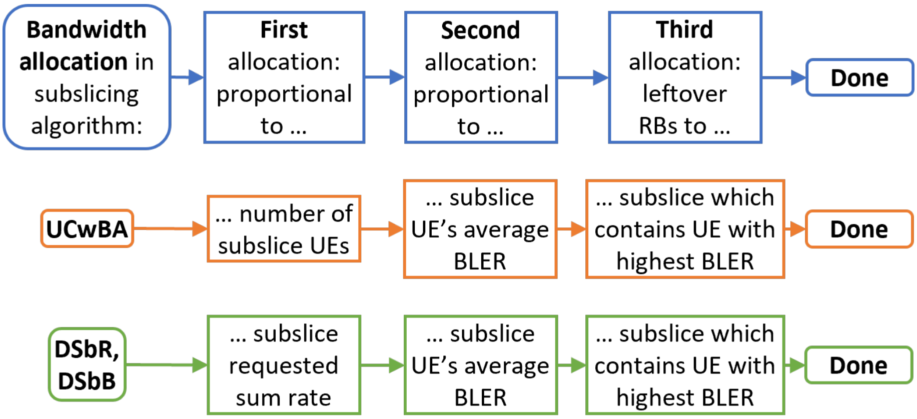

Bandwidth allocation in subslicing algorithm:
First allocation: proportional to ...
Second allocation: proportional to ...
Third allocation: leftover RBs to ...
Done
UCwBA
... number of subslice UEs
... subslice UE's average BLER
... subslice which contains UE with highest BLER
Done
DSbR, DSbB
... subslice requested sum rate
... subslice UE's average BLER
... subslice which contains UE with highest BLER
Done

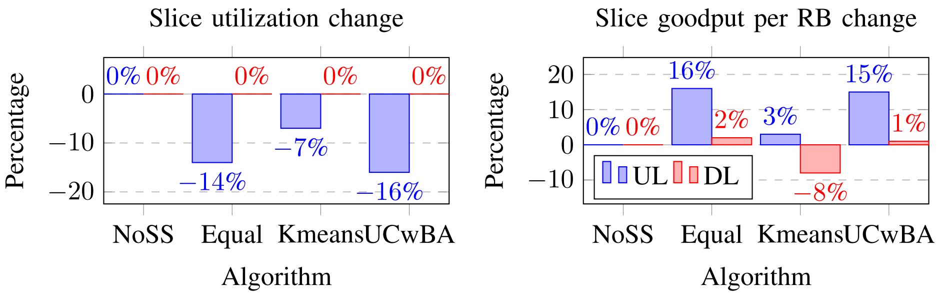
Figure 12: Bandwidth allocation to subslices in different proposed subslicing algorithms.

4.4 Evaluation of the Subslicing Algorithms

The slice performance evaluated by collecting values of 6 KPIs: utilization of UL (utilUL) and DL (utilDL), goodput per one RB in UL (gdp1UL) and DL (gdp1DL), and average BLER in UL (blerUL) and DL (blerDL). The slice utilization is calculated as a mean of all subslice utilizations. The slice goodput per one allocated RB is calculated by a sum goodput of all subslices divided by the number of allocated RBs to the slice. The slice BLER is calculated as a mean of subslice BLERs which are means of subslice UE BLERs.

The detailed results of slice performance evaluation with subslicing is in my publications 2 [2] and 4 [4]. The following subsections contain some excerpts of the most notable results.

4.4.1 Effect of the Minimum Subslice Size



(a) Slice utilization change *(b) Slice goodput per RB change*

Figure 13: Slice performance change compared if no subslicing (NoSS), equal sized subslices (Equal), UEs clustered using k-means, and proposed subslicing algorithm UCwBA from my publication 2 [2] were used. The slice contained all good-BLER UEs and UE packet size was 40 B.

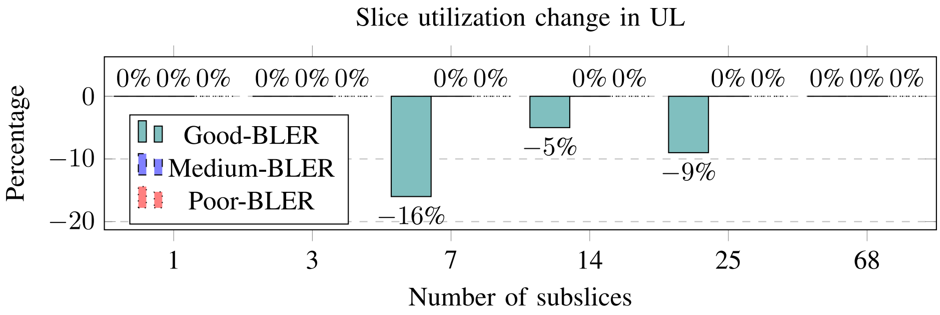
Let us consider the minimum subslice size requirement 37 RBs which is a start of best utilization zone with reasonable goodput from the Chapter 3. If the slice size is 275 RBs, then with this constraint the number of subslices could be $275/37 = 7.43 \approx 7$. The subslice sizes of different algorithms are shown in Tab. 6. If equal grouping subslicing algorithm is used then the average subslice size is $275/7 = 39.3 \approx 39$ RBs. The slice consists of 5 subslices in size of 39 RBs and 2 subslices in size of 40 RBs. The algorithm called K-means has no limit on cluster size, and UEs are clustered by UE BLER. The slice consists of the smallest subslice in size of 21 RBs, and the largest subslice in size of 65 RBs. The algorithm called user clustering with bandwidth allocation (UCwBA) from my publication 2 [2] has UEs clustered by UE BLER and the minimum subslice size constraint is 37 RBs. The slice which is subsliced using UCwBA consists of subslices in size of in the range of 37-43 RBs.

Algorithm	**Range of sizes of subslices**
Equal grouping: subslice sizes as equal as possible	39-40 RBs
K-means: UEs into subslices by UE similarity	21-65 RBs
UCwBA: uses minimum subslice size constraint 37 RBs	37-43 RBs

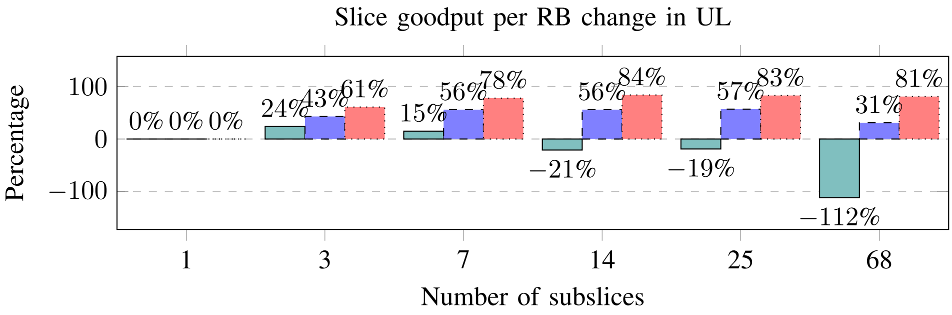
Table 6: Sizes of subslices created by different subslicing algorithms.

In Fig. 13 the slice performance is compared, if it is subsliced into 7 subslices by using algorithms equal grouping (Equal), UE clustering using k-means, and UCwBA. The utilization has decreased in UL by 7% if k-means was used, and twice as more if subslices were not smaller than 37 RBs. The utilization decrease is the greatest, 14% or 16% in UL if equal grouping or UCwBA algorithms were used, respectively. The goodput increase is the greatest in UL, 16% and 15% if equal grouping and UCwBA algorithms were used, respectively.

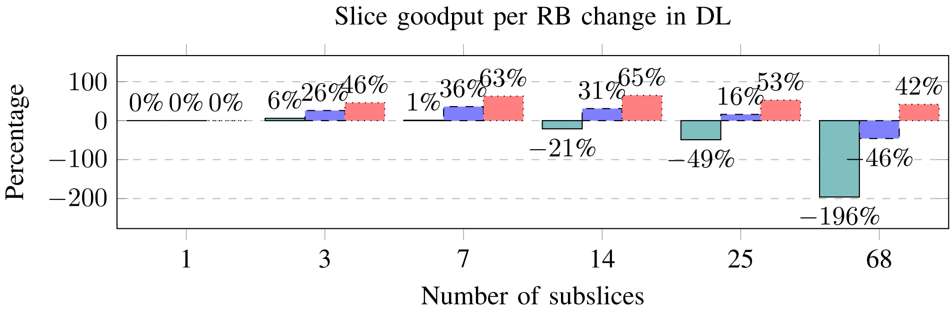
4.4.2 Effect of Subslicing for Slice Containing Set of UEs With Different BLERs



(a) Slice utilization change in UL



(b) Slice goodput per RB change in UL



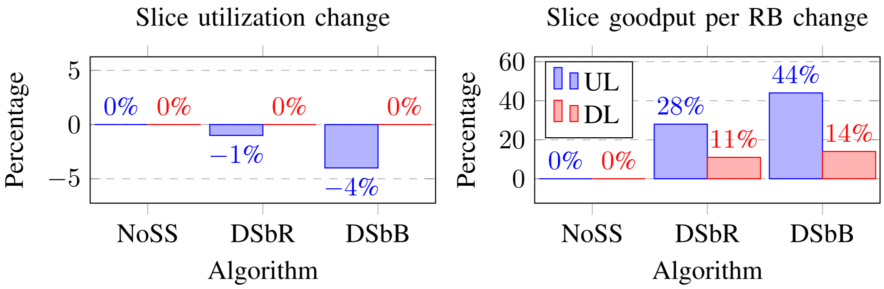
(c) Slice goodput per RB change in DL

Figure 14: Slice performance change compared if slice contained all good-BLER, medium-BLER or poor-BLER UEs and UE packet size was 40 B. The proposed subslicing algorithm UCwBA from my publication 2 [2] was used.

Let us consider sets of slice UEs with different BLER. The good-BLER, medium-BLER and poor-BLER slice is a slice which contains all UEs with good, medium or poor BLERs, respectively. The slice size is 275 RBs and all UEs request equal rates. The performance of the slice is evaluated if it is subsliced into 3, 7, 14, 25 or 68 subslices with the minimum subslice size requirement of 73, 37, 19, 11 and 4 RBs, respectively, for subslicing algorithm UCwBA from my publication 2 [2].

Slice simulation results are shown in Fig. 14. The slice utilization in UL was decreased only if the slice contained only good-BLER UEs up to 16%. The slice utilization in DL did not change and always the value of 0.998 was achieved. The slice goodput increased for good-BLER slice the most, 24% in UL, if it contained 3 subslices. The slice goodput for medium-BLER slice increased 56% in UL and 36% in DL, if it contained 7 subslices. For poor-BLER slice, the best goodput increase was 84% and 63% in UL and DL, respectively, if it contained 14 subslices. Thus, the worse BLER, the smaller subslices will improve slice performance. The goodput improvement is greater when slice contains the set of UEs with worse BLER.

4.4.3 Effect of UE Clustering by Rate and BLER



(a) Slice utilization change

(b) Slice goodput per RB change

Figure 15: Slice performance change compared if no subslicing (NoSS), UEs clustered by requested rate (DSbR) and UEs clustered by achieved BLER (DSbB). Algorithms are from my publication 4 [4].

Let us consider the slice with size of 250 RBs containing UEs with different BLERs and requested rates of 100 kbps, 200 kbps or 400 kbps from my publication 4 [4]. In this publication, the subslicing works as follows: slice is split recursively until no split required for subslice. The subslicing algorithm called dynamic subslicing by rate (DSbR) did split the slice and further split one of the subslices resulting to 3 subslices. Subslice 1 contained UEs requesting 100 kbps, subslice 2 contained UEs requesting 200 kbps and subslice 3 contained UEs requesting 400 kbps. The subslicing algorithm called dynamic subslicing by BLER (DSbB) did split the slice, then split one of the subslices and then split one of its subslices resulting to 4 subslices. Subslice 1 contained good-BLER UEs and was the largest, subslices 2 and 3 contained medium-BLER UEs, and subslices 4 and 5 were the smallest and contained poor-BLER UEs.

The slice simulation results are shown in Fig. 15. The slice performance is compared if it was subsliced using DSbR and DSbB algorithms. The former subslicing algorithm clusters UEs by their requested rate, the latter clusters UEs by their achieved BLER. The slice utilization in UL has decreased up to 4% and in DL it did not change. The slice goodput in UL has increased by 28% and 44% if algorithms DSbR and DSbB were used, respectively. The slice goodput increase in DL was lower.

4.5 Conclusion

The subslicing algorithm contains three steps:

1. conversion of the minimum subslice size requirement for the UE clustering algorithm;
2. UE clustering;
3. RB allocation to UE clusters.

The minimum subslice size requirement prevents too small subslices by setting an additional constraint to the clustering algorithm. The clustering algorithm fills all required clusters with required number of UEs, otherwise the subslices are too small because too few RBs are allocated to the UE cluster. The bandwidth allocation runs in 3 steps. First, the majority of the RBs are allocated proportional to the number of UEs or subslice requested sum rate. Secondly, minority of the RBs are allocated proportional to group BLER. Finally, the leftover RBs are allocated, which may remain due to rounding in previous steps.

The slice simulation results show that subslicing improves slice performance the most by goodput increase in both UL and DL, and smaller utilization decrease is achieved in UL. The application of the minimum subslice size criterium in subslicing reduces the slice utilization an additional 7-8 % and increases the slice goodput in UL three times. If the slice contains UEs with other than good-BLER, then the subslicing can increase slice goodput up to 61% and 84% in UL for medium-BLER slice and poor-BLER slice, respectively. The increase in DL slice goodput is up to 36% and 65% for medium-BLER and poor-BLER slice, respectively. The peak increase values were achieved in slice containing more subslices if UE BLER was worse. The subslicing with UE clustering by achieved BLER reduces slice utilization in UL 3% more than UE clustering by requested rate. The increase in slice goodput was 16% greater, if subslicing contained UE clustering by achieved BLER than requested rate.

5 Enhanced Decision Mechanisms for RAN Subslicing

Based on subslice performance, a decision is needed for a subslice: whether it should be split, merged or not changed. The subslice performance data that are used in this chapter, are Dataset 1 described in Section 3.2.1 or Dataset 2 described in Section 3.2.2.

5.1 Training Data Labeling

The inputs are values of six KPIs: bandwidth utilization in UL (utilUL) and DL (utilDL), slice goodput per one RB in UL (gdp1UL) and DL (gdp1DL), BLER in UL (blerUL), and DL (blerDL). The output is the decisions for subslice: "split", "merge" or "no change".

The output decision "split" is for subslice which is too large and splitting into two can improve its performance. The subslice splitting is described in Chapter 6. The decision "merge" is for subslice which is too small and needs to be merged with another too small subslice. The subslice merging is described in Chapter 6. The decision "no change" is for subslice which has suitable size and performance.

For Dataset 1, performance data clustering is proposed, and for Dataset 2, the optimization problem is solved for data labeling.

5.1.1 Performance Data Clustering

Let us use the Dataset 1, described in Section 3.2.1, which contains slice performance results at subslice sizes 4-275 RBs and packet sizes were different in five subsets.There are three decisions; therefore, three clusters are necessary. Based on the results presented in my publication 3 [3] the performance data contains all 6 KPIs and is clustered using k-means into 3 clusters. The clustering result is shown in Fig. 16a.

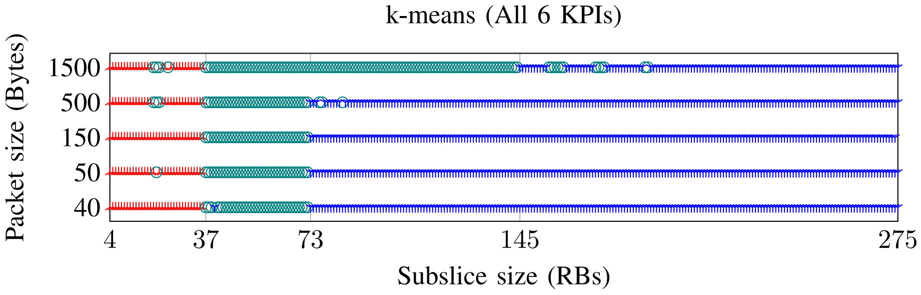
5.1.2 Discussion of Deciding Subslice Operation by Clustering Result

The decision method described in the previous subsection uses subslice performance data if subslice contained all good-BLER UEs.

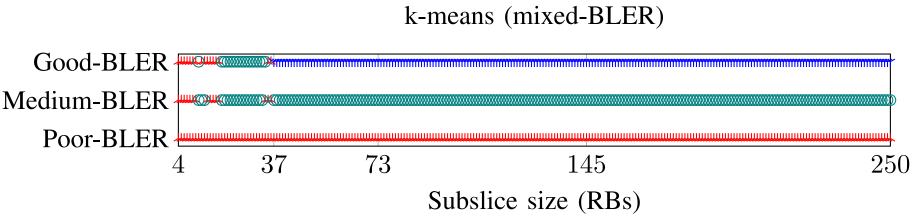
Now, I attempt to apply this method to Dataset 2, described in Section 3.2.2, which contains slice performance results at subslice sizes of 4-250 RBs, and UE BLERs were different in three subsets. In the subsets, the subslice contains all good-BLER, all medium-BLER or all poor-BLER UEs. All three subsets were clustered by six KPIs together, and clustering results are shown in Fig. 16b. The subset containing good-BLER UEs contained data in three clusters, where too small and too large subslices can be distinguished. The subset containing medium-BLER UEs contained data in two clusters ("merge" and "no change"), and the subset containing poor-BLER UEs contained data in one cluster ("merge"). This shows that subslices containing poor-BLER UEs should be merged; however, my simulations results in my publication 2 [2] have shown that for poor-BLER UEs, better slice performance is achieved if there are many small subslices.

If clustering each BLER subset separately, then the results are shown in Fig. 16c. The clustering results do not depend on UE BLER; however, the performance of a subsliced slice was better if poor-BLER UEs could be in smaller subslices as the results in my publication 2 [2] had shown.

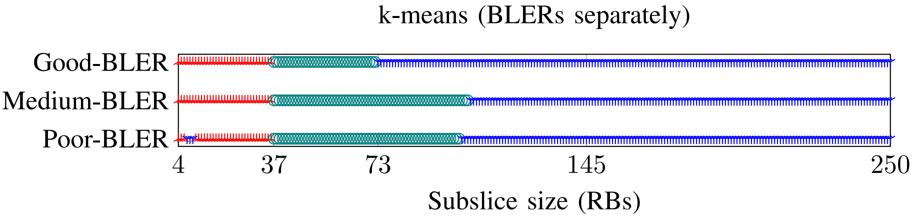
The conclusion is that the best subslice sizes should be found for each BLER subset separately, they are different and cannot be found by clustering the performance data.



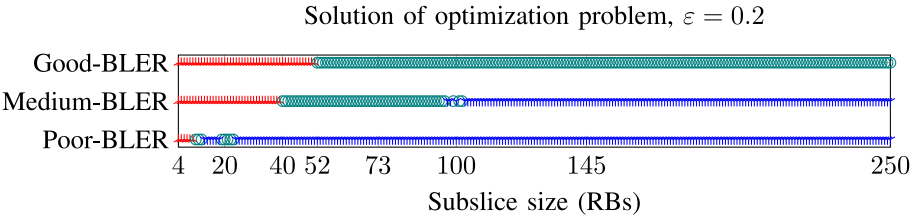
(a) All good-BLER UEs



(b) UEs with different BLERs clustered together



(c) UEs with different BLERs clustered separately



(d) Solution of optimization problem, $\varepsilon = 0.2$

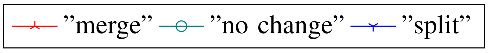
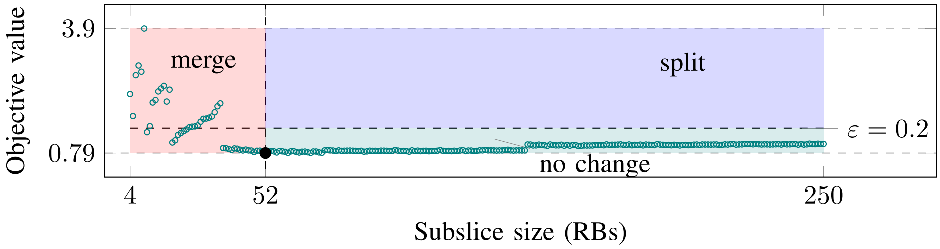


Figure 16: Training data obtained by (a), (b), (c) performance data clustering and (d) solving an optimization problem.

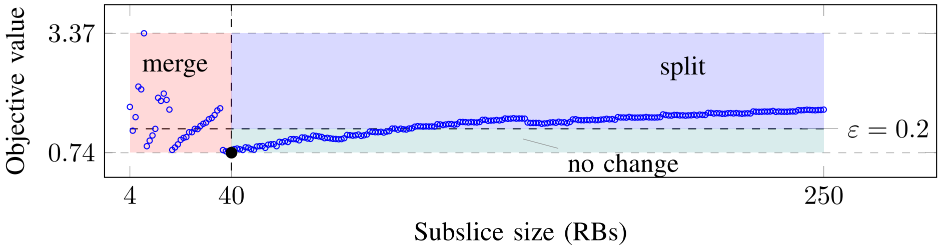
5.1.3 Optimization Problem to Find Best Subslice Size

Let us use the Dataset 2 which was described in Section 3.2.2 (three sets of UEs - good-BLER, medium-BLER, and poor-BLER; subslice size 4-250 RBs, packet size 40 B) from my publication 4 [4].

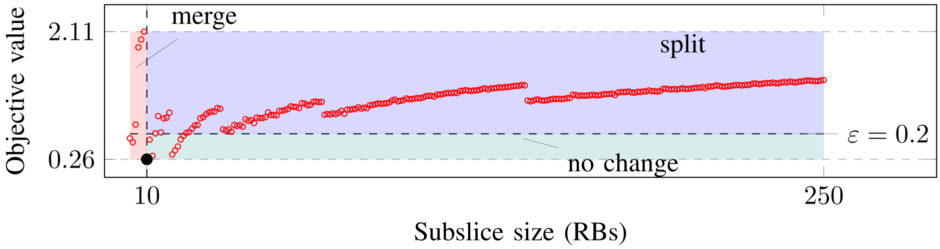
The optimization problem was solved for each BLER subset separately. Objective function defined as root-mean-square error (RMSE) of KPI value differences from best values of the subset. The best subslice size is when the objective function has the minimum value. In Fig 17 it can be seen that the best subslice sizes are 52, 40 and 10 RBs for good-BLER, medium-BLER and poor-BLER UEs, respectively.



(a) Good-BLER



(b) Medium-BLER



(c) Poor-BLER

○ Good-BLER ○ Medium-BLER ○ Poor-BLER

Figure 17: Results of optimization problem of each BLER subset, from my publication 4 [4].

The ε-neighbourhood of sizes which had objective value close to the best objective value were decided as "no change". The greater ε value results more subslice sizes to be in suitable size, thus less slice reconfiguration operations needed. On the other hand, insufficient number of subslice splitting cannot reach subslices of the best size and too many splits can create too small subslices. Simulation results of initialization shown in [4], that $\varepsilon = 0.1$ was too small and created too many and too small subslices while $\varepsilon = 0.2$ created less subslices and dynamic subslicing algorithms created slice which outperformed the slice of static subslicing algorithm from [84].

5.2 Neural Network and Search of Hyperparameters

The machine learning tools suitable for multi-class classification are k nearest neighbour (KNN), supporting vector machine (SVM) and neural network (NN). A neural network was selected as the tool for classification. NN has the flexibility and can work with large datasets.

The decision mechanism contains classifier NN, which is shown in Fig. 18. Both in my publication 3 [3] and 4 [4] the fully connected feedforward classifier NN were trained to learn the decisions of subslice operation. Number of hidden layers and neurons (nodes) on hidden layer was selected by trials considering 1-2 layers and 2-22 neurons. Neurons used the rectified linear unit (ReLU) activation function.

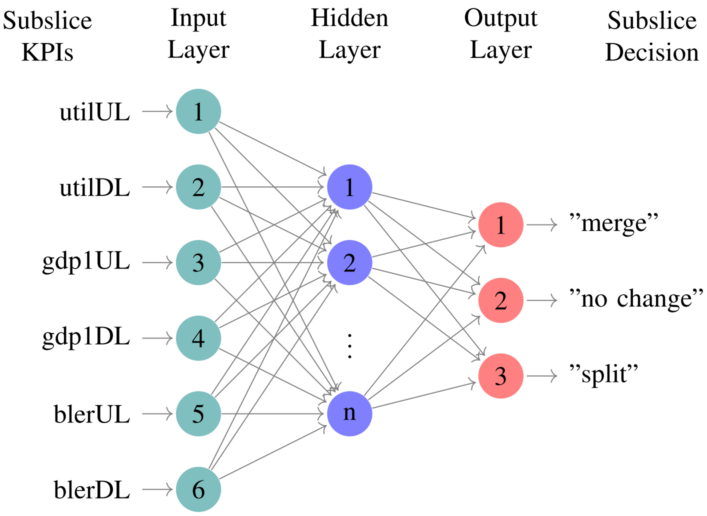


Figure 18: Classifier NN for subslice decisions.

Considering cross entropy losses in graphs, it was visually detected the minimum number of neurons where the losses did not increase further to avoid overfit. The selected hidden layer configuration consists of one hidden layer consisting of 7 neurons and 9 neurons, for training data shown in Fig. 16a and 16d, respectively.

5.3 Conclusion

In this chapter, the performance data labeling methods for subslice size simulation results are proposed whether the slice or subslice should be split, merged or not changed. The performance data clustering did not work if the slice contained UEs with different BLERs. The optimization problem to find best subslice sizes was proposed to solve separately for each BLER group dataset - if slice contains all good-BLER, medium-BLER and poor-BLER UEs, respectively. The objective value is calculated as the sum of RMSE of differences between the best value of KPI and current value of KPI.

The NN has the values of six KPIs as inputs and the output is the decision "split", "merge" or "no change". The NN is used in the management CCL for subslicing which is described in the next chapter.

6 Subslicing in Closed Control Loop to Improve Slice Performance

In this chapter, it is discussed how subslicing can be used automatically when it is able to improve slice performance without additional bandwidth allocation. The proposed MCCL uses the MAPE-K [118] architecture. It has the goal to improve slice performance on fixed bandwidth by subslicing. It uses the decision mechanism proposed in Chapter 5 and subslicing algorithms proposed in Chapter 4. The proposed MCCL implementation for subslicing is published in my publication 4 [4].

6.1 Management Closed Control Loop Architecture

In the next subsections, the 4 functions of MAPE-K and a database are described. The used architecture is shown in Fig. 19. The goal of the MCCL is to improve slice performance by subslicing.

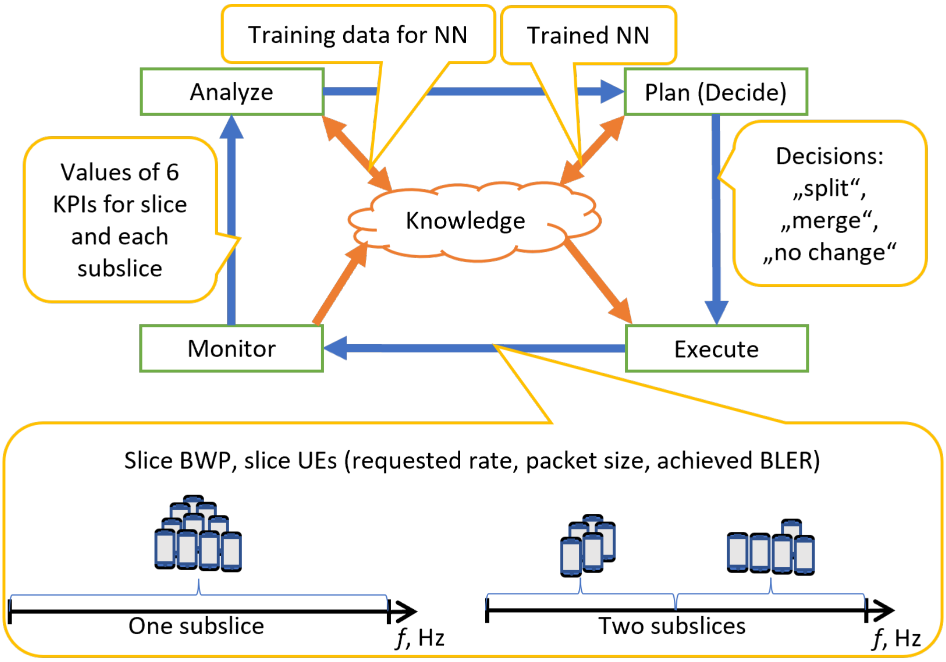

Training data for NN
Trained NN
Analyze
Plan (Decide)
Values of 6 KPIs for slice and each subslice
Knowledge
Decisions: „split“, „merge“, „no change“
Monitor
Execute
Slice BWP, slice UEs (requested rate, packet size, achieved BLER)
One subslice
f, Hz
Two subslices
f, Hz

Figure 19: Management closed control loop architecture

6.1.1 Monitor

The monitor function collects slice data: slice BWP size, number of UEs, slice requested sum rate, subslices and their settings, and 6 KPIs. It calculates averages for KPI values per time period. In [66], the length of time period is recommended 30 sec. For UEs, the requested rate and achieved BLER are collected, and to which subslice it belongs.

The monitor function produces the knowledge about slice performance.

6.1.2 Analyze

Analyze function creates training data for neural network (NN). Analyze function consumes knowledge about slice performance and produces the knowledge about labeling the slice performance data whether the slice or subslice should be split, merged or not changed.

The NN is trained using slice KPI values. This can be done periodically or on request. It is desired to retrain NN when the slice performance does not improve by subslicing, but as this takes time and energy, then retraining should be done not too frequently.

6.1.3 Decide (Plan)

The Decide function uses trained NN to decide the operation for each subslice in the slice, whether it should be split, merged or not changed.

6.1.4 Execute

The execute function processes the slice and subslice decisions. First, the subslices with "no change" decision are processed, followed by "merge" and "split".

Subslice merging creates an ordered list of subslices with the decision "merge". If odd number of subslices the smallest 3 are merged, else the smallest is merged with the largest until all subslices are merged pairwise. The sets of UEs and number for RBs are combined.

Subslice splitting is done one-by one for each subslice to be split. subslicing algorithms are used for subslicing into 2 subslices.

6.2 Evaluation

The slice configuration is initialized as shown in my publication 4 [4]. I compare the slice performance in proposed MCCL if slice is split using three algorithms: static 3 subslices (S3S), dynamic subslicing by rate (DSbR) and dynamic subslicing by BLER (DSbB). The first is adopted from [84] and the latter two subslice splitting algorithms are proposed in my publication 4 [4].

6.2.1 Simulation Setup

The slice BWP is n28 (FDD, UL 730-748 MHz, DL 758-803 MHz) which is a 5G-NR band specified in 3GPP TS 38.104 [9]. The subcarrier spacing is 15 kHz, thus maximum subslice bandwidth is 250 RBs.

The slice UEs are the set of 250 UEs and their details are shown in the Tab. 7. The UE BLERs and requested rates are both combinations of different 3 values. This set of UEs is expected to achieve the similar performance, as the dataset 2 which was described in Section 3.2.2. Both the slice in this section and the subslice of its maximum size in dataset 2 had 250 UEs and sum rate is 50 Mbps.

6.2.2 Scenarios

First, the subslicing initialization is done. This means that the slice will reach to the stable configuration where all subslices have a decision "no change". When the initialization is done, then the scenario starts.

Two runtime scenarios are considered: traffic increase and traffic decrease. Traffic change can be implemented by addition or removal of UEs. The greatest effect is when the UE requested rate and achieved BLER are both high.

Table 7: The combinations of three requested rates and three different BLERs for a set of 250 UEs in a slice, which requested sum rate is 50 Mbps.

BLER	**Requested rate**	**Number of UEs**
Any	100 kbps	110
Any	200 kbps	85
Any	400 kbps	55
Total	**50 Mbps**	**250**
Good-BLER	100 or 200 or 400 kbps	83
Medium-BLER	100 or 200 or 400 kbps	84
Poor-BLER	100 or 200 or 400 kbps	83
Total	**50 Mbps**	**250**
Good-BLER	100 kbps	36
Good-BLER	200 kbps	28
Good-BLER	400 kbps	19
Medium-BLER	100 kbps	37
Medium-BLER	200 kbps	29
Medium-BLER	400 kbps	18
Poor-BLER	100 kbps	37
Poor-BLER	200 kbps	28
Poor-BLER	400 kbps	18
Total	**50 Mbps**	**250**

The traffic increase scenario is implemented for these simulations such that the 5 poor-BLER UEs which request 400 kbps rates each, are admitted to the slice. The slice requested sum rate increases at each such event by 2 Mbps (+4%). The UEs are admitted to the subslice with the least bandwidth utilization. Similarly, the traffic decrease is implemented for these simulations that 5 UEs, which request 400 kbps rate are removed from their respective subslices. The leaving UEs have achieved worst BLER values in the last simulation. The slice requested sum rate decreases at each such event by 2 Mbps (-4%). The slice should adapt to the change of requested sum rate by the reconfiguration of subslicing which achieves the best slice performance.

The workflow of the discrete event simulator is shown in Fig. 20. In initialization, the event does not happen and second simulation of the slice is not needed. In scenarios 1 and 2 if the subslicing configuration did not change then the second run of slice is skipped. The run MCCL means that the mean values of KPIs are the input of decision NN which decides for each subslice in the slice whether it needs to be split, merged, or not changed. Next, the execute function processes the subslices and executes the decisions. The output of MCCL is the new slice configuration regarding subslicing - which UE belongs to which subslice and how many RBs are allocated for each subslice in the slice.

The slice is simulated 10 times and KPIs are calculated as mean values with error bars in the graph showing confidence interval of 95%.

6.2.3 Results of Initialization

The detailed slice performance data and configuration regarding subslicing at the end of initialization is shown in my publication 4 [4]. The Algorithm S3S created three subslices because the set of UEs had three different requested rates. DSbR algorithm also created three subslices after two MCCL runs. The difference between S3S and DSbR is that if UEs

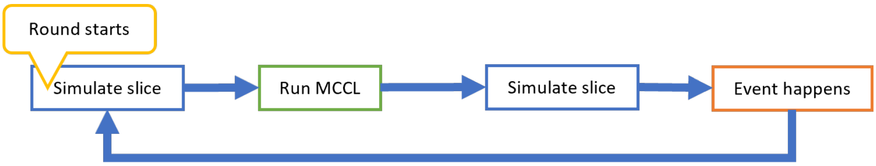


Figure 20: Discrete event simulator for evaluation of the slice performance during the scenario.

are added or removed, then S3S adjusts the RB allocation. DSbR does not adjust RB allocation, if the decision for subslice is "no change”. Algorithm DSbB created three subslices for good- and medium-BLER UEs. For poor-BLER UEs, one subslice has an infinite split-merge loop: in one round, the splitting is required, and in the next round, the splits need to be merged. The initialization results are the starting point for runtime scenarios, of which simulation results are presented in the next subsections.

6.2.4 Results of Runtime, Scenario 1

The slice performance results of scenario 1 are shown in Fig. 21 left column. The slice requested sum rate will increase by 4% at each event, reaching to 44% increase at the last event.

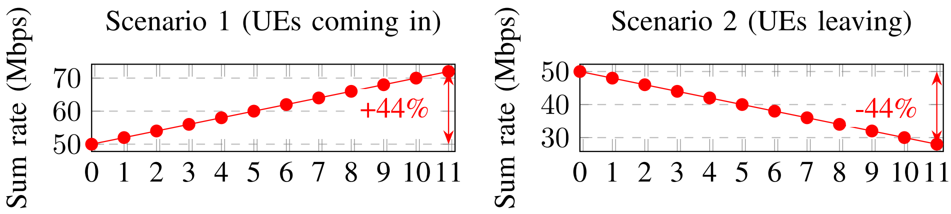
The utilization in UL and DL remains full (100%) if no subslicing. The splitting algorithms S3S and DSbR reach to 100% in UL utilization after first and second event. For DSbB 100% utilization is reached in UL at event 5 and in DL at event 2. The goodput per RB in UL and DL slowly decrease. The best goodput per one RB in UL is achieved if the splitting algorithm is DSbB. In DL initially S3S and DSbR achieve best goodput per RB, however after event 2 DSbB has the highest goodput per RB in DL. When the splitting algorithm DSbR was used, the 3 subslices worked until event 3 (see Fig. 23), when the subslice with $id = 3$ was decided to be split. Later on, after the next event, the one subslice needs to be merged, but it is a single subslice with decision of merge. This affected BLER, which decreased in both UL and DL. The slice configuration if splitting algorithm S3S was used has always 3 subslices and minor RB adjustments proportional to subslice sum rate (see Fig. 22). When the splitting algorithm DSbB was used, then the two poor-BLER subslices with $id = 8$ and $id = 9$ are split and merged in the cycle (see Fig. 24). When there are total six subslices in the slice at event 2 and 5, then the goodput and BLER values are lower and when the slice consists of four subslices, then the goodput and BLER values are higher. This split-merge cycle ends after event 7 when there are two subslices, each in size of 20 RBs for poor-BLER UEs. The performance of those subslices balance on the boundary of decisions of "no change" and "merge".

At the end of scenario 1 the utilization is 100%, best goodput per on RB is achieved by using splitting algorithm DSbB, lowest BLER is in UL with DSbB and in DL with DSbR. No subslicing had the worst values for all six KPIs.

6.2.5 Results of Runtime, Scenario 2

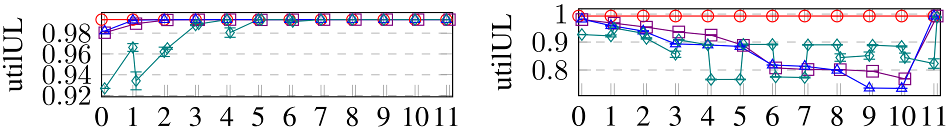
The slice performance results are shown in Fig. 21 right column. The slice requested sum rate will decrease by 4% at each event, reaching to 44% decrease at the last event.

The decrease of requested sum rate has the following effects for slice KPIs. If no subslicing then the utilization does not change. If subslice splitting algorithms are used, then UL utilization decreases. The utilization of a slice, if splitting algorithm DSbB was used, depends on the number of poor-BLER subslices.



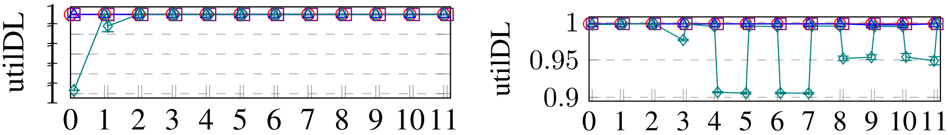
(a) Requested sum rate, scenario 1

(b) Requested sum rate, scenario 2



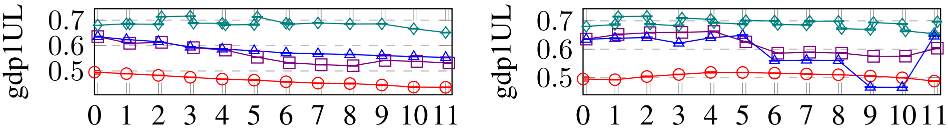
(c) Bandwidth utilization in uplink (utilUL), scenario 1

(d) Bandwidth utilization in uplink (utilUL), scenario 2



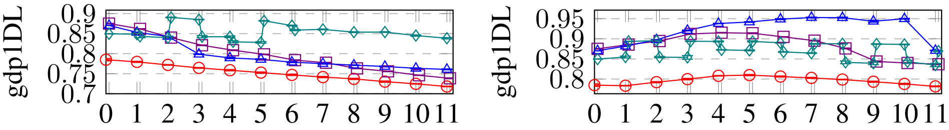
(e) Bandwidth utilization in downlink (utilDL), scenario 1

(f) Bandwidth utilization in downlink (utilDL), scenario 2



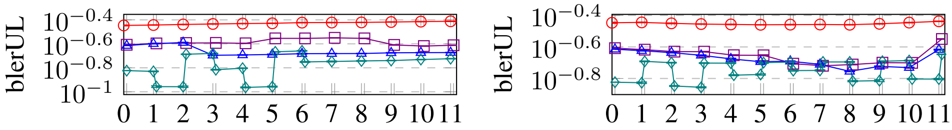
(g) Goodput per one RB in uplink (gdp1UL), scenario 1

(h) Goodput per one RB in uplink (gdp1UL), scenario 2



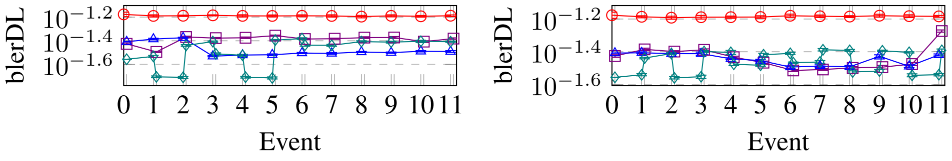
(i) Goodput per one RB in downlink (gdp1DL), scenario 1

(j) Goodput per one RB in downlink (gdp1DL), scenario 2



(k) Slice BLER in uplink (blerUL), scenario 1

(l) Slice BLER in uplink (blerUL), scenario 2



(m) Slice BLER in downlink (blerDL), scenario 1

(n) Slice BLER in downlink (blerDL), scenario 2

○ No subslicing □ S3S △ DSbR ◇ DSbB

Figure 21: Slice performance runtime. (a), (c), (e), (g), (i), (k), (m) scenario 1. (b), (d), (f), (h), (j), (l), (n) scenario 2.

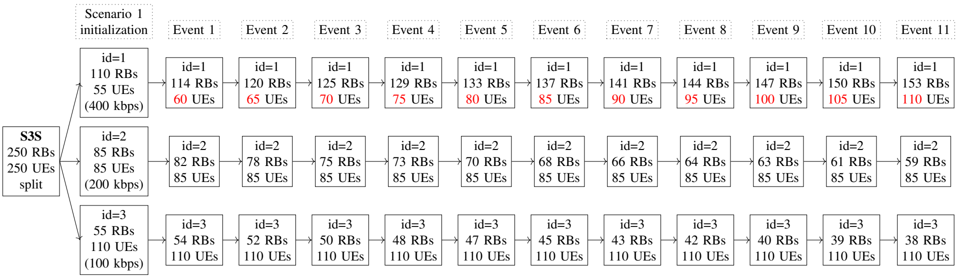


Figure 22: Slice configuration, scenario 1, splitting algorithm S3S.

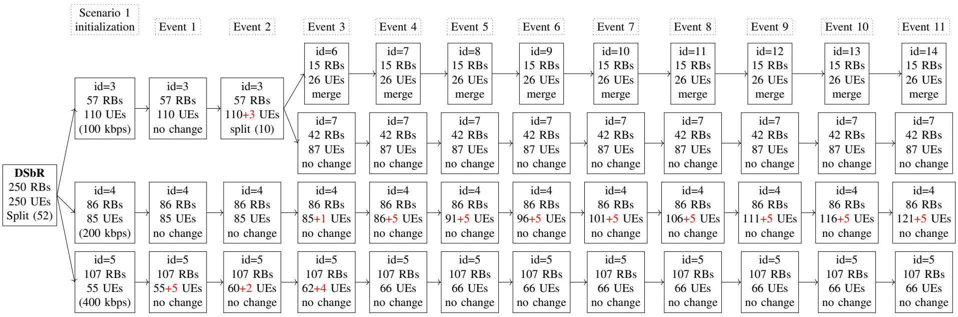

Scenario 1 initialization
Event 1
Event 2
Event 3
Event 4
Event 5
Event 6
Event 7
Event 8
Event 9
Event 10
Event 11
DSbR 250 RBs 250 UEs Split (52)
id=6 15 RBs 26 UEs merge
id=7 15 RBs 26 UEs merge
id=8 15 RBs 26 UEs merge
id=9 15 RBs 26 UEs merge
id=10 15 RBs 26 UEs merge
id=11 15 RBs 26 UEs merge
id=12 15 RBs 26 UEs merge
id=13 15 RBs 26 UEs merge
id=14 15 RBs 26 UEs merge
id=3 57 RBs 110 UEs (100 kbps)
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110+3 UEs split (10)
id=7 42 RBs 87 UEs no change
id=7 42 RBs 87 UEs no change
id=7 42 RBs 87 UEs no change
id=7 42 RBs 87 UEs no change
id=7 42 RBs 87 UEs no change
id=7 42 RBs 87 UEs no change
id=7 42 RBs 87 UEs no change
id=7 42 RBs 87 UEs no change
id=7 42 RBs 87 UEs no change
id=4 86 RBs 85 UEs (200 kbps)
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85+1 UEs no change
id=4 86 RBs 86+5 UEs no change
id=4 86 RBs 91+5 UEs no change
id=4 86 RBs 96+5 UEs no change
id=4 86 RBs 101+5 UEs no change
id=4 86 RBs 106+5 UEs no change
id=4 86 RBs 111+5 UEs no change
id=4 86 RBs 116+5 UEs no change
id=4 86 RBs 121+5 UEs no change
id=5 107 RBs 55 UEs (400 kbps)
id=5 107 RBs 55+5 UEs no change
id=5 107 RBs 60+2 UEs no change
id=5 107 RBs 62+4 UEs no change
id=5 107 RBs 66 UEs no change
id=5 107 RBs 66 UEs no change
id=5 107 RBs 66 UEs no change
id=5 107 RBs 66 UEs no change
id=5 107 RBs 66 UEs no change
id=5 107 RBs 66 UEs no change
id=5 107 RBs 66 UEs no change
id=5 107 RBs 66 UEs no change

Figure 23: Slice configuration, scenario 1, splitting algorithm DSbR.

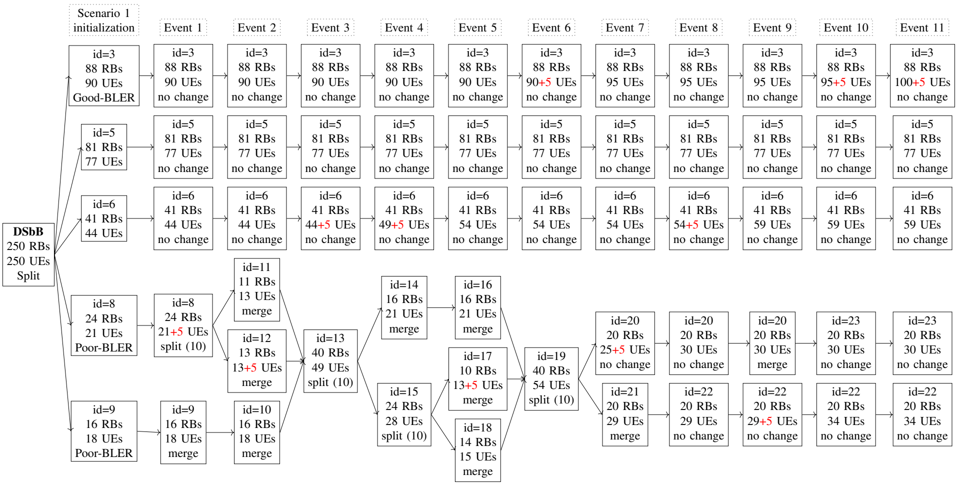

Scenario 1 initialization
Event 1
Event 2
Event 3
Event 4
Event 5
Event 6
Event 7
Event 8
Event 9
Event 10
Event 11
DSbB 250 RBs 250 UEs Split
id=3 88 RBs 90 UEs Good-BLER
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90+5 UEs no change
id=3 88 RBs 95 UEs no change
id=3 88 RBs 95 UEs no change
id=3 88 RBs 95 UEs no change
id=3 88 RBs 95+5 UEs no change
id=3 88 RBs 100+5 UEs no change
id=5 81 RBs 77 UEs
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=6 41 RBs 44 UEs
id=6 41 RBs 44 UEs no change
id=6 41 RBs 44 UEs no change
id=6 41 RBs 44+5 UEs no change
id=6 41 RBs 49+5 UEs no change
id=6 41 RBs 54 UEs no change
id=6 41 RBs 54 UEs no change
id=6 41 RBs 54 UEs no change
id=6 41 RBs 54+5 UEs no change
id=6 41 RBs 59 UEs no change
id=6 41 RBs 59 UEs no change
id=6 41 RBs 59 UEs no change
id=8 24 RBs 21 UEs Poor-BLER
id=8 24 RBs 21+5 UEs split (10)
id=11 11 RBs 13 UEs merge
id=12 13 RBs 13+5 UEs merge
id=13 40 RBs 49 UEs split (10)
id=14 16 RBs 21 UEs merge
id=16 16 RBs 21 UEs merge
id=15 24 RBs 28 UEs split (10)
id=17 10 RBs 13+5 UEs merge
id=18 14 RBs 15 UEs merge
id=19 40 RBs 54 UEs split (10)
id=20 20 RBs 25+5 UEs no change
id=20 20 RBs 30 UEs no change
id=20 20 RBs 30 UEs merge
id=23 20 RBs 30 UEs no change
id=23 20 RBs 30 UEs no change
id=21 20 RBs 29 UEs merge
id=22 20 RBs 29 UEs no change
id=22 20 RBs 29+5 UEs no change
id=22 20 RBs 34 UEs no change
id=22 20 RBs 34 UEs no change
id=9 16 RBs 18 UEs Poor-BLER
id=9 16 RBs 18 UEs merge
id=10 16 RBs 18 UEs merge

Figure 24: Slice configuration, scenario 1, splitting algorithm DSbB.

At the end of this scenario, all UEs with requested rate of 400 kbps have left. The S3S splitting algorithm achieves 100% utilization again. This happened, because the subslice, which contained UEs which request 400 kbps, had earlier low utilization and now become empty (see Fig. 25). The subslice with $id = 1$ had sufficient resources for good performance before. The splitting algorithm DSbR achieved at the end of the scenario 100% utilization, but the cause is different (see Fig. 26. The 107 RBs, which were allocated to subslice with $id = 10$, are freed from the slice.

The goodput per RB in UL initially increases and then decreases. Initial increase means that when some UEs have left the slice, then the free RBs enabled additional transmissions. Later on, after event 5, the UEs do not have enough data to be transmitted. If subslice splitting algorithm DSbR was used, then at event 9 the goodput in UL is lower than when no subslicing done. This is caused by the MCCL algorithm where if subslice has decision "no change" then its number of RBs is not changed (see Fig. 26). The UEs are leaving one of the subslices and this subslice has resource surplus. The splitting algorithm DSbB changes poor-BLER subslice configuration (see Fig. 27). If more subslices then lower utilization, goodput and BLER achieved if this subslice splitting is used. The lowest BLER is when splitting algorithm DSbB was used; however, depending on the number of subslices for poor-BLER UEs the slice BLER was comparable to that achieved by using other subslice splitting algorithms.

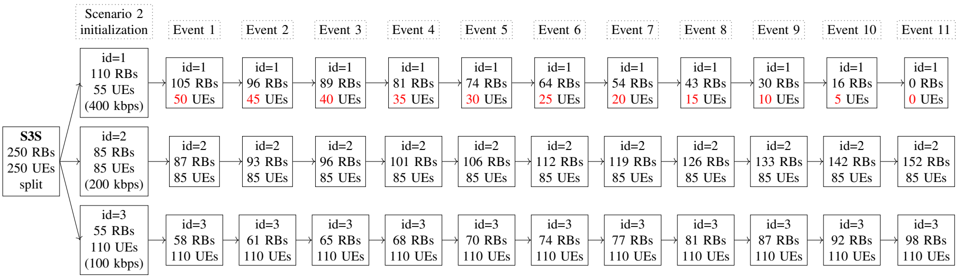


Figure 25: slice configuration, scenario 2, splitting algorithm S3S.

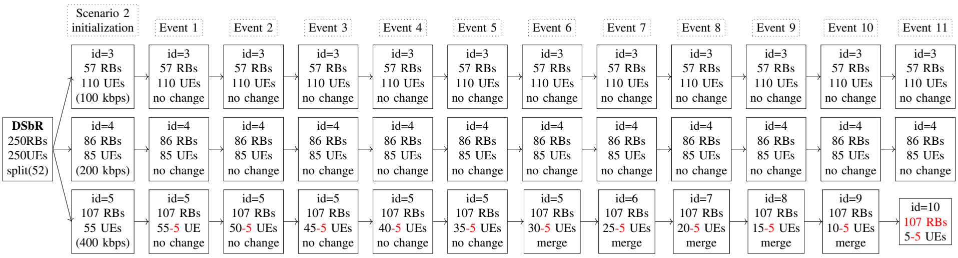

Scenario 2 initialization
Event 1
Event 2
Event 3
Event 4
Event 5
Event 6
Event 7
Event 8
Event 9
Event 10
Event 11
DSbR 250RBs 250UEs split(52)
id=3 57 RBs 110 UEs (100 kbps)
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=4 86 RBs 85 UEs (200 kbps)
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=5 107 RBs 55 UEs (400 kbps)
id=5 107 RBs 55-5 UE no change
id=5 107 RBs 50-5 UEs no change
id=5 107 RBs 45-5 UEs no change
id=5 107 RBs 40-5 UEs no change
id=5 107 RBs 35-5 UEs no change
id=5 107 RBs 30-5 UEs merge
id=6 107 RBs 25-5 UEs merge
id=7 107 RBs 20-5 UEs merge
id=8 107 RBs 15-5 UEs merge
id=9 107 RBs 10-5 UEs merge
id=10 107 RBs 5-5 UEs

Figure 26: slice configuration, scenario 2, splitting algorithm DSbR.

6.2.6 Discussion of Possible Improvements

The subslice merging process needs improvement on the case when just one subslice needs to be merged. In current MCCL implementation, the single subslice which requires merging, is waiting for another subslice which requires merging. The another subslice to be merged will appear on next MCCL run as if subslice splitting algorithm DSbB was used. If subslice splitting algorithm DSbR was used, then the single subslice to be merged waits merging for a long time. The better decision for the single subslice requiring "merge" would be merging this subslice with the smallest subslice which had decision "no change". This is a potential configuration conflict and needs a tradeoff between not merging the subslice, which requires to be merged, and changing the subslice, which does not require its configuration change.

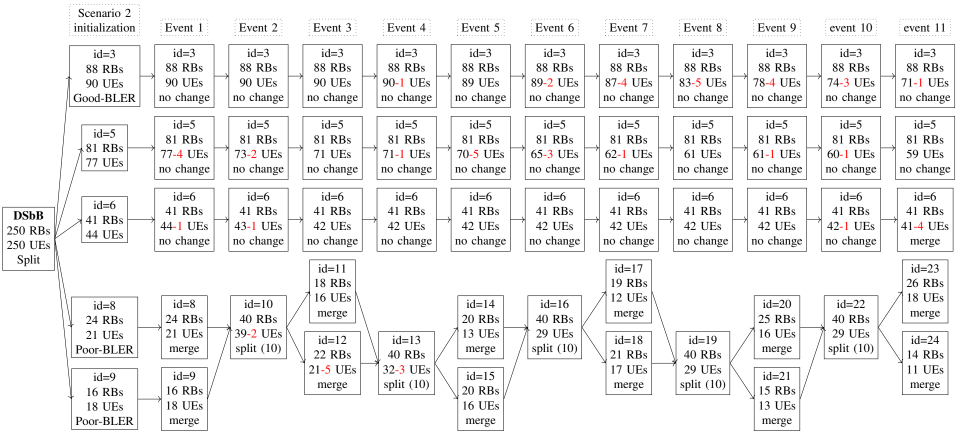

Scenario 2 initialization
Event 1
Event 2
Event 3
Event 4
Event 5
Event 6
Event 7
Event 8
Event 9
event 10
event 11
DSbB 250 RBs 250 UEs Split
id=3 88 RBs 90 UEs Good-BLER
id=5 81 RBs 77 UEs
id=6 41 RBs 44 UEs
id=8 24 RBs 21 UEs Poor-BLER
id=9 16 RBs 18 UEs Poor-BLER
id=3 88 RBs 90 UEs no change
id=5 81 RBs 77-4 UEs no change
id=6 41 RBs 44-1 UEs no change
id=8 24 RBs 21 UEs merge
id=9 16 RBs 18 UEs merge
id=3 88 RBs 90 UEs no change
id=5 81 RBs 73-2 UEs no change
id=6 41 RBs 43-1 UEs no change
id=10 40 RBs 39-2 UEs split (10)
id=3 88 RBs 90 UEs no change
id=5 81 RBs 71 UEs no change
id=6 41 RBs 42 UEs no change
id=11 18 RBs 16 UEs merge
id=12 22 RBs 21-5 UEs merge
id=3 88 RBs 90-1 UEs no change
id=5 81 RBs 71-1 UEs no change
id=6 41 RBs 42 UEs no change
id=13 40 RBs 32-3 UEs split (10)
id=3 88 RBs 89 UEs no change
id=5 81 RBs 70-5 UEs no change
id=6 41 RBs 42 UEs no change
id=14 20 RBs 13 UEs merge
id=15 20 RBs 16 UEs merge
id=3 88 RBs 89-2 UEs no change
id=5 81 RBs 65-3 UEs no change
id=6 41 RBs 42 UEs no change
id=16 40 RBs 29 UEs split (10)
id=3 88 RBs 87-4 UEs no change
id=5 81 RBs 62-1 UEs no change
id=6 41 RBs 42 UEs no change
id=17 19 RBs 12 UEs merge
id=18 21 RBs 17 UEs merge
id=3 88 RBs 83-5 UEs no change
id=5 81 RBs 61 UEs no change
id=6 41 RBs 42 UEs no change
id=19 40 RBs 29 UEs split (10)
id=3 88 RBs 78-4 UEs no change
id=5 81 RBs 61-1 UEs no change
id=6 41 RBs 42 UEs no change
id=20 25 RBs 16 UEs merge
id=21 15 RBs 13 UEs merge
id=3 88 RBs 74-3 UEs no change
id=5 81 RBs 60-1 UEs no change
id=6 41 RBs 42-1 UEs no change
id=22 40 RBs 29 UEs split (10)
id=3 88 RBs 71-1 UEs no change
id=5 81 RBs 59 UEs no change
id=6 41 RBs 41-4 UEs merge
id=23 26 RBs 18 UEs merge
id=24 14 RBs 11 UEs merge

Figure 27: Slice configuration, scenario 2, splitting algorithm DSbB.

Another idea for improvement is the training data labeling for the NN. The boundaries between decision "merge" and "no change" could be adjusted such that all subslices which objective value is within ε-neighborhood (see Fig. 17) could have the decision "no change" regardless of the subslice size greater or less than the best subslice size. Currently, all subslices with the size smaller than best subslice size are labelled "merge".

The first improvement idea avoids the subslices with "merge" decision left unchanged, and the second improvement avoids "merge" decisions if the performance can match "no change" decision.

6.3 Conclusion

In this chapter, the MCCL implementation was proposed, which improves slice performance by subslicing. The decision function used the proposed decision mechanism presented in Section 5.1.3 and the execution function used the proposed subslice splitting algorithm DSbB.

The subslicing improves slice performance in MCCL. The splitting algorithms, which cluster UEs by requested rate (S3S, DSbR), had similar slice performance and did not change slice configuration frequently. If DSbB, which clusters UEs by UE BLER, was used for slice splitting, the slice had lower bandwidth utilization and BLER, and higher goodput per one RB than other splitting algorithms. However, slice reconfiguration was done at each MCCL run. Not all subslices were reconfigured, but only those which contained UEs with poor BLER. This is caused by training data labeling, where poor-BLER dataset had very few "no change" decisions. The size of ε-neighbourhood was the same for each BLER dataset, but it seems to be too small for poor-BLER dataset.

If having training data for both good-BLER subslices and subslices with worse BLER, the slices containing mixed-BLER UEs can be subsliced in MCCL with the slice performance improvement.

7 Conclusion and Future Works

This chapter summarizes the research presented in this thesis, provides answers to research questions, and suggests future research directions.

7.1 Summary

In this thesis, the RAN slice performance dependence from the RAN slice size was investigated. Based on the slice performance dependence on slice size, the subslicing was proposed. Moreover, an automatic subslicing was implemented in the management closed control loop with the goal of RAN slice performance improvement on fixed slice bandwidth.

RAN slice was simulated using MATLAB version R2021b 5G toolbox tool called NR Cell performance evaluation with physical layer integration. Contrary to common research, where the set of UEs are modelled, in this work for each RB allocated it was consumed by a UE by its requested rate in both UL and DL. This created a close-to slice overload situation. The RAN slice performance was evaluated in UL and DL by bandwidth utilization, goodput per one allocated RB, and average UE BLER.

First, the slice performance dependence on slice size was discovered. Based on this RAN slice performance data, the RAN subslicing was proposed to improve slice performance on fixed slice bandwidth. RAN subslicing means that slice UEs are grouped, and slice bandwidth divided to the UE groups. Slice performance improvement means that bandwidth utilization has decreased, goodput per one RB increased and slice average UE BLER decreased.

Secondly, based on the slice performance dependence on the slice size, the subslicing algorithms were proposed. The higher slice performance improvement can be achieved if UEs were clustered by BLER than clustered by requested rate. One constraint to subslicing is that subslices should not be too small. The minimum subslice size depends on the average BLER of UE group. Simulation results show the minimum subslice size could be 52, 40 and 10 RBs for good-BLER, medium-BLER and poor-BLER UE group, respectively.

Thirdly, the decision mechanism based on subslice performance for subslice splitting or merging is proposed, which works for UEs with any BLER. The training dataset for NN should contain training examples of KPIs for both good-BLER subslices and poor-BLER subslices. The data labeling for decisions should be done separately for both datasets.

Fourthly, the subslicing algorithms and decision mechanism are building blocks of management CCL for automatic subslicing with the goal to improve slice performance on fixed bandwidth by subslicing.

7.2 Research Questions Answered

The research questions are answered based on this thesis as follows.

RQ1: How many subslices can be in the RAN slice? The maximum number of subslices in RAN slice depends on slice performance. The subslices must not be too small, because too small subslices have low goodput per one allocated RB. This deteriorates RAN slice performance. RAN subslicing can improve the slice performance in a close-to-slice overload situation. If the slice contains all good-BLER UEs, the number of subslices can be a few. That is to 2-5 subslices, because the solution of the optimization problem in Section 5.1.3 is 52 RBs for the best subslice size, and the number of subslices depends on the size of the slice BWP. For poor-BLER UEs the RAN slice needs more subslices for slice performance improvement on fixed slice bandwidth.

RQ2: How much radio resource can be allocated to the slice and subslice to achieve the best performance? The sufficient radio resource should be allocated to the RAN slice and subslice, and efficient resource allocation leads to the close-to-slice-overload situation. However, the BWP size allocated to the RAN slice or subslice affects performance. Based on my simulation results of 5G-NR, there can be found a minimum subslice size limit of 37 RBs. Too small BWPs exhibit poor performance, except if BLER of all UEs in the subslice is other than good.

RQ3: How users should be clustered, and slice bandwidth be allocated to subslices to improve RAN slice performance? In this thesis, the UE requested rate and achieved BLER were considered. In the research, the UE clustering by their SLA is proposed, as the service verticals were created. This enables to share multiple different types of resources efficiently. In RAN, the resource is the bandwidth in number of RBs. The 5G-NR simulation results in MATLAB R2021b show that the UE clustering by their achieved BLER resulted in lower utilization in UL, higher goodput and lower slice BLER than UE clustering by their requested rate. Subslicing does not create more bandwidth. It uses the performance dependence on subslice size. RB allocation should be proportional to requested sum rate and group BLER. Therefore, subslices with poor BLER need more RBs, but too many RBs cannot be taken away from with good BLER.

RQ4: What is the impact of RAN subslicing under the management closed control loop framework? The MCCL framework can be used to improve slice performance if the dependence between slice size and performance exists and the decision mechanism has matching training data. The training dataset needs to have samples of sets of UEs of different BLERs. The UE BLER has been shown to affect the RAN slice performance dependence from the slice size. Therefore, subslicing by BLER into subslices with different sizes has been shown to improve slice performance the most. The subslice splitting into more than two at the time requires more work on slice reconfiguration if one of the subslices require configuration change (merge or split) later. Therefore, the subslice splitting into two is executed for subslices which require smaller size for improved performance. Decision boundaries affect how much reconfiguration is done and if the best number and subslice sizes are achieved.

7.3 Contributions and Limitations

This work has contributed to the RAN slicing as follows:

- **How to find RAN slice performance dependence on its BWP size:** The performance was evaluated at all possible slice sizes in granularity of one RB. For this reason, the considered requested rates should be able to consume the RB. The most straightforward approach is to map one UE per one RB.

  Limitation: current work has not considered a situation if performance data is missing for one or multiple subslice sizes.

- **How different slice/UE requirements and capabilities affect RAN slice performance dependence on slice size:** Training data should contain samples if slice contains UEs with different requirements and capabilities, that are possible in actual slice.

  Limitation: current work has considered only packet sizes and UE BLERs as UE requirements and capabilities.

- **How to find the subslice size with its best performance:** the current KPI values are compared with the best achieved KPI values. If the sum difference is small, then this subslice size is better.

Limitation: current work has considered bandwidth utilization, goodput per one allocated RB and average BLER of all UEs; other KPIs are not studied.

- **How to decide the subslice operations (increase/decrease its size):** If subslice size must be increased, then another subslice should be found. If subslice size must be decreased, then the maximum sizes of splits are when subslice is split into two.

  Limitation: if subslice needs to be split, then it was not investigated whether the performance would be better when it should be split into more than two. It was assumed that if split into more than two will improve performance more, it will be done in multiple MCCL runs.

- **Performance data labeling for decisions:** The slice performance pattern is plotted in slice size (x) and objective value (y) plane. If there is one minimum, then the horizontal boundary determined by the ε-neighbourhood distinguishes subslices, which need changes in their configuration and which do not. The vertical boundary distinguishes subslices to be changed, into split or merge decisions.

  Limitation: current work covers situation when only one minimum value for an objective function exists.

- **When to start/stop the MCCL:** The slice performance was investigated in close-to slice overload conditions.

  Limitation: it is assumed that RAN slice bandwidth resources are sufficient and not excessive.

7.4 Future Work

Based on the limitations described in the previous section, future research directions can be proposed:

- **Training data collection for classifier NN to create decisions for subslices:** In this thesis, the subslicing solutions were proposed based on a complete simulated RAN slice performance dataset, i.e. performance data exists for each possible BWP size and fixed rate requirements per one RB. In actual RAN slice, the performance data collection at each possible BWP size is not possible. It needs the further investigation what data could be used for training the NN which decides subslice configuration changes.
- **Investigation of dependencies between subslice size and UE requirements and capabilities:** Currently, the effect of UE BLER to the best subslice size was noticed. Subslicing by UE BLER improves RAN slice performance more than subslicing by UE rate. For poor-BLER UEs the smaller subslices exhibit better performance. It needs further investigation in actual RAN slice if there are more dependencies between UE requirements/capabilities and better subslice sizes and how to use them for RAN slice performance improvement by subslicing.
- **Proposal of new KPIs:** Currently, the RAN slice performance improvement is measured by increase in goodput per one RB, while bandwidth utilization and average UE BLER decrease. The latency is a critical requirement to satisfy, and it needs further research how latency can be affected by RAN slice BWP resizing.
- **Training data labeling:** If it is discovered in RAN that multiple BWP sizes are optimal, then how to choose between them and how to reconfigure subslices of other sizes.

List of Figures

1 Architecture of 5GS 14
2 MCCL and thesis organization.... 18
3 Base station decomposition - CUPS and functional splits. (a) possible decomposition of a BS, (b) BS split and cloudifying by 3GPP and used in 3G/4G RAN, (c) CUPS and CU-DU split used in 5G RAN, (d) SD-RAN split into 3 units by O-RAN, (e) possible physical placement of BS components 20
4 Slice LCM example in case of slice consists of one CN slice subnet and one RAN slice subnet.... 23
5 The constituents of resource block according to 3GPP TS 38.211 [35]. On the time dimension, it contains the 12 or 14 symbols that form a slot. On the frequency dimension, it contains 12 subcarriers that form a resource block. 25
6 Simplified scheme of inter- and intra-slice scheduling of latency (L-slice), fixed (M-slice), throughput (T-slice) and best effort (B-slice). Request or packet notation contains a letter to identify the slice and number to identify UE in that slice. The figure shows a snapshot at a given time instant. ... 27
7 Types of simulators.... 38
8 Subslice size simulation results. Left column contains dataset 1 from my publication 3 [3], right column contains dataset 2 from my publication 4 [4]. 45
9 Performance zones based on slice size and performance. 46
10 Ranges of KPI values in percentages in different performance zones of subslices in all possible sizes. 47
11 An example of how the proposed clustering algorithm clusters data points into three clusters with a minimum cluster size of 2. The data points cluster well into two clusters (green and orange). Figure from my publication 2 [2]. 51
12 Bandwidth allocation to subslices in different proposed subslicing algorithms. 51
13 Slice performance change compared if no subslicing (NoSS), equal sized subslices (Equal), UEs clustered using k-means, and proposed subslicing algorithm UCwBA from my publication 2 [2] were used. The slice contained all good-BLER UEs and UE packet size was 40 B. 52
14 Slice performance change compared if slice contained all good-BLER, medium-BLER or poor-BLER UEs and UE packet size was 40 B. The proposed subslicing algorithm UCwBA from my publication 2 [2] was used. 53
15 Slice performance change compared if no subslicing (NoSS), UEs clustered by requested rate (DSbR) and UEs clustered by achieved BLER (DSbB). Algorithms are from my publication 4 [4]. 54
16 Training data obtained by (a), (b), (c) performance data clustering and (d) solving an optimization problem. 57
17 Results of optimization problem of each BLER subset, from my publication 4 [4]. 58
18 Classifier NN for subslice decisions. 59
19 Management closed control loop architecture 60
20 Discrete event simulator for evaluation of the slice performance during the scenario. 63
21 Slice performance runtime. (a), (c), (e), (g), (i), (k), (m) scenario 1. (b), (d), (f), (h), (j), (l), (n) scenario 2. 64
22 Slice configuration, scenario 1, splitting algorithm S3S. 65

23 Slice configuration, scenario 1, splitting algorithm DSbR. 65
24 Slice configuration, scenario 1, splitting algorithm DSbB. 65
25 slice configuration, scenario 2, splitting algorithm S3S. 66
26 slice configuration, scenario 2, splitting algorithm DSbR. 66
27 Slice configuration, scenario 2, splitting algorithm DSbB. 67

List of Tables

2 Schedulers comparison 29
3 5G RAN emulators 37
4 Simulators 40
5 MATLAB Toolbox settings 44
6 Sizes of subslices created by different subslicing algorithms. 52
7 The combinations of three requested rates and three different BLERs for a set of 250 UEs in a slice, which requested sum rate is 50 Mbps. 62

References

[1] M. Kulmar, I. Muursepp, and M. M. Alam, "The Impact of RAN Slice Bandwidth Subpartitioning on Slice Performance," in *2022 Int. Wirel. Commun. Mob. Comput.*, IEEE, May 2022, pp. 419–424, ISBN: 978-1-6654-6749-0, DOI: `10.1109/IWCMC55113.2022.9825262`.

[2] M. Kulmar, I. Müürsepp, and M. M. Alam, "Heuristic Radio Access Network Subslicing with User Clustering and Bandwidth Subpartitioning," *Sensors*, vol. 23, no. 10, p. 4613, May 2023, ISSN: 1424-8220, DOI: `10.3390/s23104613`.

[3] M. Kulmar, I. Müürsepp, and M. M. Alam, "Enhanced Decision Mechanism for RAN Subslicing in Management Closed Control Loop," pp. 175–181, Sep. 2023, DOI: `10.1109/FMEC59375.2023.10306223`.

[4] M. Kulmar, M. M. Alam, and I. Müürsepp, "Management Closed Control Loop Automation for Improved RAN Slice Performance by Subslicing," *TechRxiv. December 18*, 2023, Submitted to IEEE Open J. Commun. Soc. (under review), DOI: `10.22541/techrxiv.170290948.88447519/v1`.

[5] P. Jonsson, Ed., *Ericsson Mobility Report June 2022*, 2023, [Online]. Available: `https://www.ericsson.com/en/reports-and-papers/mobility-report`.

[6] ITU-R, "IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond. ITU-R M.2083," Tech. Rep., 2015, [Online]. Available: `https://www.itu.int/rec/R-REC-M.2083-0-201509-I/en`.

[7] ITU-R, "M2410 - Minimum requirements related to technical performance for IMT-2020 radio interface(s)," Tech. Rep. Report ITU-R M.2410-0, 2017, pp. 1–11, [Online]. Available: `https://www.itu.int/pub/R-REP-M.2410`.

[8] M. Johansson J, *Network slicing: the top 10 most profitable industries*, 2021, [Online]. Available: `https://www.ericsson.com/en/blog/2021/6/network-slicing-top-10-industries` (visited on 11/08/2023).

[9] 3GPP, "TS 38.104 - NR; Base Station (BS) radio transmission and reception," 2022, [Online]. Available: `https://www.3gpp.org/DynaReport/38104.htm`.

[10] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Commun. Surv. Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013, ISSN: 1553-877X, DOI: `10.1109/SURV.2013.013013.00155`.

[11] C. Mei, J. Liu, J. Li, L. Zhang, and M. Shao, "5G network slices embedding with sharable virtual network functions," *J. Commun. Networks*, vol. 22, no. 5, pp. 415–427, Oct. 2020, ISSN: 1229-2370, DOI: `10.1109/JCN.2020.000026`.

[12] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Perez, "Multi-Tenant Radio Access Network Slicing: Statistical Multiplexing of Spatial Loads," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3044–3058, Oct. 2017, ISSN: 1063-6692, DOI: `10.1109/TNET.2017.2720668`, arXiv: `1607.08271`.

[13] S. D'Oro, F. Restuccia, A. Talamonti, and T. Melodia, "The Slice Is Served: Enforcing Radio Access Network Slicing in Virtualized 5G Systems," *Proc. - IEEE INFOCOM*, vol. 2019-April, pp. 442–450, 2019, ISSN: 0743166X, DOI: `10.1109/INFOCOM.2019.8737481`.

[14] 3GPP, "TS 28.530 - Management and orchestration; Concepts, use cases and requirements," Tech. Rep., 2020, [Online]. Available: `https://www.3gpp.org/DynaReport/28530.htm`.

[15] 3GPP, "TS 23.501 - System architecture for the 5G System (5GS); Stage 2," Tech. Rep., 2023, [Online]. Available: `https://www.3gpp.org/DynaReport/23501.htm`.

[16] MathWorks, *NR Cell Performance Evaluation with Physical Layer Integration*, 2022, [Online]. Available: `https://se.mathworks.com/help/5g/ug/nr-cell-performance-evaluation-with-physical-layer-integration.html` (visited on 01/19/2022).

[17] ETSI, "GS NFV-EVE 003 V1.1.1 Network Functions Virtualisation (NFV); Ecosystem; Report on NFVI Node Physical Architecture Guidelines for Multi-Vendor Environment.," Tech. Rep., 2016, [Online]. Available: `https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports`.

[18] ETSI, "GS NFV-MAN 001 V1.1.1. Network Functions Virtualisation (NFV); Management and Orchestration," Tech. Rep., 2014, [Online]. Available: `https://portal.etsi.org/webapp/workprogram/Report_WorkItem.asp?WKI_ID=41954`.

[19] ETSI, "GS NFV-INF 003 - V1.1.1 - Network Functions Virtualisation (NFV); Infrastructure; Compute Domain," Tech. Rep., 2014, pp. 1–57, [Online]. Available: `http://webapp.etsi.org/workprogram/Report_WorkItem.asp?WKI_ID=42377`.

[20] A. Lara, A. Kolasani, and B. Ramamurthy, "Network Innovation using OpenFlow: A Survey," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 493–512, 2014, ISSN: 1553-877X, DOI: `10.1109/SURV.2013.081313.00105`.

[21] 3GPP, "Study on new radio access technology: Radio access architecture and interfaces (Release 14) TR 38.801 V14.0.0 (2017-03)," 3GPP, Tech. Rep., 2017, [Online]. Available: `https://www.3gpp.org/DynaReport/38801.htm`.

[22] O-RAN, "O-RAN Architecture Description. O-RAN.WG1.OAD-R003-v09.00," Tech. Rep., 2023, [Online]. Available: `https://orandownloadsweb.azurewebsites.net/specifications`.

[23] O-RAN, "O-RAN.WG1.Slicing-Architecture-R003-v10.00," Tech. Rep., 2023, pp. 1–70, [Online]. Available: `https://orandownloadsweb.azurewebsites.net/specifications`.

[24] GSMA, "Generic Network Slice Template Version 2.0," Tech. Rep., 2019, pp. 1–61, [Online]. Available: `https://www.gsma.com/newsroom/all-documents/generic-network-slice-template-v2-0/`.

[25] GSMA, "NG . 116 Generic Network Slice Template 27 April 2023," no. April, pp. 1–71, 2023, [Online]. Available: `https://www.gsma.com/newsroom/resources/ng-116-generic-network-slice-template-v9-0/`.

[26] 3GPP, "TS 38.300 NR; NR and NG-RAN Overall Description; Stage 2," 2021, [Online]. Available: `https://www.3gpp.org/DynaReport/38300.htm`.

[27] 3GPP, "TS 23.003 - Numbering, addressing and identification," Tech. Rep., 2021, [Online]. Available: `https://www.3gpp.org/DynaReport/23003.htm`.

[28] 3GPP, "TS 29.531 - 5G System; Network Slice Selection Services; Stage 3," 2020, [Online]. Available: `https://www.3gpp.org/DynaReport/29531.htm`.

[29] 3GPP, "TS 28.533 - Management and orchestration; Architecture framework," 2020, [Online]. Available: `https://www.3gpp.org/DynaReport/28533.htm`.

[30] ETSI, “GR NFV-EVE 012 V3.1.1. Network Functions Virtualisation (NFV) Release 3 ; Evolution and Ecosystem ; Report on Network Slicing Support with ETSI NFV Architecture Framework.,” Tech. Rep., 2017, [Online]. Available: `https://portal.etsi.org/webapp/workprogram/Report_WorkItem.asp?WKI_ID=51391`.

[31] 3GPP, “TS 28.535 - Management and orchestration; Management services for communication service assurance; Requirements,” Tech. Rep., 2020, [Online]. Available: `https://www.3gpp.org/DynaReport/28535.htm`.

[32] 3GPP, “TS 28.541 - Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3,” Tech. Rep., 2021, [Online]. Available: `https://www.3gpp.org/DynaReport/28541.htm`.

[33] 3GPP, “TS 28.500 Telecommunication management; Management concept, architecture and requirements for mobile networks that include virtualized network functions,” 2020, [Online]. Available: `https://www.3gpp.org/DynaReport/28500.htm`.

[34] ETSI, *Open Source MANO*, 2021, [Online]. Available: `https://osm.etsi.org/`.

[35] 3GPP, “TS 38.211 - NR; Physical channels and modulation,” 2020, [Online]. Available: `https://www.3gpp.org/DynaReport/38211.htm`.

[36] ETSI, “ETSI GS ZSM 009-1: Zero-Touch Network and Service Management (ZSM); Closed-loop automation; Enablers,” Tech. Rep., 2021, pp. 1–37, [Online]. Available: `http://webapp.etsi.org/workprogram/Report_WorkItem.asp?WKI_ID=58053`.

[37] ETSI, “Experiential Networked Intelligence (ENI); Overview of Prominent Control Loop Architectures,” vol. 1, pp. 1–17, 2021, [Online]. Available: `https://www.etsi.org/deliver/etsi_gr/ENI/001_099/017/02.01.01_60/gr_ENI017v020101p.pdf`.

[38] I. Vaishnavi and L. Ciavaglia, “Challenges Towards Automation of Live Telco Network Management: Closed Control Loops,” in *2020 16th Int. Conf. Netw. Serv. Manag.*, IEEE, Nov. 2020, pp. 1–5, ISBN: 978-3-903176-31-7, DOI: `10.23919/CNSM50824.2020.9269048`.

[39] M. Gramaglia, M. Kajo, C. Mannweiler, Ö. Bulakci, and Q. Wei, “A unified service-based capability exposure framework for closed-loop network automation,” *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 11, Nov. 2022, ISSN: 2161-3915, DOI: `10.1002/ett.4598`.

[40] M. A. Mehaseb, Y. Gadallah, A. Elhamy, and H. Elhennawy, “Classification of LTE Uplink Scheduling Techniques: An M2M Perspective,” *IEEE Commun. Surv. Tutorials*, vol. 18, no. 2, pp. 1310–1335, 2016, ISSN: 1553-877X, DOI: `10.1109/COMST.2015.2504182`.

[41] F. Capozzi, G. Piro, L. Grieco, G. Boggia, and P. Camarda, “Downlink Packet Scheduling in LTE Cellular Networks: Key Design Issues and a Survey,” *IEEE Commun. Surv. Tutorials*, vol. 15, no. 2, pp. 678–700, 2013, ISSN: 1553-877X, DOI: `10.1109/SURV.2012.060912.00100`.

[42] S. Mandelli, M. Andrews, S. Borst, and S. Klein, “Satisfying Network Slicing Constraints via 5G MAC Scheduling,” in *IEEE INFOCOM 2019 - IEEE Conf. Comput. Commun.*, IEEE, Apr. 2019, pp. 2332–2340, ISBN: 978-1-7281-0515-4, DOI: `10.1109/INFOCOM.2019.8737604`.

[43] R. Schmidt, C.-Y. Chang, and N. Nikaein, "Slice Scheduling with QoS-Guarantee Towards 5G," in *2019 IEEE Glob. Commun. Conf.*, IEEE, Dec. 2019, pp. 1–7, ISBN: 978-1-7281-0962-6, DOI: `10.1109/GLOBECOM38437.2019.9013258`.

[44] G. Zhou, L. Zhao, K. Liang, G. Zheng, and L. Hanzo, "Utility Analysis of Radio Access Network Slicing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 1163–1167, Jan. 2020, ISSN: 0018-9545, DOI: `10.1109/TVT.2019.2952216`.

[45] S. Bakri, P. A. Frangoudis, and A. Ksentini, "Dynamic Slicing of RAN Resources for Heterogeneous Coexisting 5G Services," in *2019 IEEE Glob. Commun. Conf.*, IEEE, Dec. 2019, pp. 1–6, ISBN: 978-1-7281-0962-6, DOI: `10.1109/GLOBECOM38437.2019.9013954`.

[46] Y. Han, X. Tao, X. Zhang, and S. Jia, "Hierarchical Resource Allocation in Multi-Service Wireless Networks with Wireless Network Virtualization," *IEEE Trans. Veh. Technol.*, pp. 1–1, 2020, ISSN: 0018-9545, DOI: `10.1109/TVT.2020.3019217`.

[47] W. Shi, J. Li, P. Yang, *et al.*, "Two-Level Soft RAN Slicing for Customized Services in 5G-and-Beyond Wireless Communications," *IEEE Trans. Ind. Informatics*, vol. 18, no. 6, pp. 4169–4179, Jun. 2022, ISSN: 1551-3203, DOI: `10.1109/TII.2021.3083579`.

[48] J. Mei, X. Wang, K. Zheng, G. Boudreau, A. B. Sediq, and H. Abou-Zeid, "Intelligent Radio Access Network Slicing for Service Provisioning in 6G: A Hierarchical Deep Reinforcement Learning Approach," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 6063–6078, Sep. 2021, ISSN: 0090-6778, DOI: `10.1109/TCOMM.2021.3090423`.

[49] H. Zhou, M. Elsayed, and M. Erol-Kantarci, "RAN Resource Slicing in 5G Using Multi-Agent Correlated Q-Learning," in *2021 IEEE 32nd Annu. Int. Symp. Pers. Indoor Mob. Radio Commun.*, IEEE, Sep. 2021, pp. 1179–1184, ISBN: 978-1-7281-7586-7, DOI: `10.1109/PIMRC50174.2021.9569358`.

[50] F. Song, J. Li, C. Ma, Y. Zhang, L. Shi, and D. N. K. Jayakody, "Dynamic Virtual Resource Allocation for 5G and Beyond Network Slicing," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 215–226, 2020, ISSN: 2644-1330, DOI: `10.1109/OJVT.2020.2990072`.

[51] Y. Chen, Y. Wang, M. Liu, J. Zhang, and L. Jiao, "Network Slicing Enabled Resource Management for Service-Oriented Ultra-Reliable and Low-Latency Vehicular Networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7847–7862, Jul. 2020, ISSN: 0018-9545, DOI: `10.1109/TVT.2020.2991723`.

[52] T. Ma, Y. Zhang, F. Wang, D. Wang, and D. Guo, "Slicing Resource Allocation for eMBB and URLLC in 5G RAN," *Wirel. Commun. Mob. Comput.*, vol. 2020, pp. 1–11, Jan. 2020, ISSN: 1530-8669, DOI: `10.1155/2020/6290375`.

[53] P. Korrai, E. Lagunas, S. K. Sharma, S. Chatzinotas, A. Bandi, and B. Ottersten, "A RAN Resource Slicing Mechanism for Multiplexing of eMBB and URLLC Services in OFDMA Based 5G Wireless Networks," *IEEE Access*, vol. 8, pp. 45 674–45 688, 2020, ISSN: 2169-3536, DOI: `10.1109/ACCESS.2020.2977773`.

[54] J. Pérez-Romero, O. Sallent, R. Ferrús, and R. Agustí, "On the configuration of radio resource management in a sliced RAN," *IEEE/IFIP Netw. Oper. Manag. Symp. Cogn. Manag. a Cyber World, NOMS 2018*, pp. 1–6, 2018, DOI: `10.1109/NOMS.2018.8406280`.

[55] T. Guo and A. Suarez, “Enabling 5G RAN Slicing With EDF Slice Scheduling,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2865–2877, Mar. 2019, ISSN: 0018-9545, DOI: `10.1109/TVT.2019.2894695`.

[56] C. Shannon, “Communication in the Presence of Noise,” *Proc. IRE*, vol. 37, no. 1, pp. 10–21, Jan. 1949, ISSN: 0096-8390, DOI: `10.1109/JRPROC.1949.232969`.

[57] R. Ferrús, O. Sallent, J. Pérez-Romero, and R. Agustí, “On the automation of RAN slicing provisioning: solution framework and applicability examples,” *Eurasip J. Wirel. Commun. Netw.*, vol. 2019, no. 1, pp. 1–12, 2019, ISSN: 16871499, DOI: `10.1186/s13638-019-1486-1`.

[58] A. Boubendir, F. Guillemin, S. Kerboeuf, B. Orlandi, F. Faucheux, and J. L. Lafragette, “Network slice life-cycle management towards automation,” *2019 IFIP/IEEE Symp. Integr. Netw. Serv. Manag. IM 2019*, pp. 709–711, 2019, [Online]. Available: `https://ieeexplore.ieee.org/document/8717901`.

[59] J. Garcia-Morales, M. C. Lucas-Estan, and J. Gozalvez, “Latency-Sensitive 5G RAN Slicing for Industry 4.0,” *IEEE Access*, vol. 7, pp. 143 139–143 159, 2019, ISSN: 21693536, DOI: `10.1109/ACCESS.2019.2944719`.

[60] Y. Katsumata, T. Shimojo, A. Khan, A. Yamada, and S. Iwashina, “An efficient cost-calculation method for end-to-end slice generation,” in *2018 15th IEEE Annu. Consum. Commun. Netw. Conf.*, IEEE, Jan. 2018, pp. 1–6, ISBN: 978-1-5386-4790-5, DOI: `10.1109/CCNC.2018.8319213`.

[61] M. R. Raza, A. Rostami, L. Wosinska, and P. Monti, “A Slice Admission Policy Based on Big Data Analytics for Multi-Tenant 5G Networks,” *J. Light. Technol.*, vol. 37, no. 7, pp. 1690–1697, Apr. 2019, ISSN: 0733-8724, DOI: `10.1109/JLT.2019.2896138`.

[62] V. Sciancalepore, X. Costa-Perez, and A. Banchs, “RL-NSB: Reinforcement Learning-Based 5G Network Slice Broker,” *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1543–1557, Aug. 2019, ISSN: 1063-6692, DOI: `10.1109/TNET.2019.2924471`.

[63] B. Han, V. Sciancalepore, X. Costa-Perez, D. Feng, and H. D. Schotten, “Multiservice-Based Network Slicing Orchestration With Impatient Tenants,” *IEEE Trans. Wirel. Commun.*, vol. 19, no. 7, pp. 5010–5024, Jul. 2020, ISSN: 1536-1276, DOI: `10.1109/TWC.2020.2988644`.

[64] F. Meneses, R. Silva, D. Corujo, and R. L. Aguiar, “Micro and Macro Network Slicing: An Experimental Assessment of the Impact of Increasing Numbers of Slices,” *Wirel. Pers. Commun.*, Mar. 2019, ISSN: 0929-6212, DOI: `10.1007/s11277-019-06267-4`.

[65] I. Afolabi, T. Taleb, P. A. Frangoudis, M. Bagaa, and A. Ksentini, “Network Slicing-Based Customization of 5G Mobile Services,” *IEEE Netw.*, vol. 33, no. 5, pp. 134–141, Sep. 2019, ISSN: 0890-8044, DOI: `10.1109/MNET.001.1800072`.

[66] S. Kuklinski and L. Tomaszewski, “Key Performance Indicators for 5G network slicing,” in *2019 IEEE Conf. Netw. Softwarization*, IEEE, Jun. 2019, pp. 464–471, ISBN: 978-1-5386-9376-6, DOI: `10.1109/NETSOFT.2019.8806692`.

[67] N. Salhab, S. E. Falou, R. Rahim, S. E. E. Ayoubi, and R. Langar, “Optimization of the implementation of network slicing in 5G RAN,” in *2018 IEEE Middle East North Africa Commun. Conf.*, IEEE, Apr. 2018, pp. 1–6, ISBN: 978-1-5386-1254-5, DOI: `10.1109/MENACOMM.2018.8371035`.

[68] C. Campolo, A. Molinaro, A. Iera, R. R. Fontes, and C. E. Rothenberg, “Towards 5G Network Slicing for the V2X Ecosystem,” in *2018 4th IEEE Conf. Netw. Softwarization Work.*, IEEE, Jun. 2018, pp. 400–405, ISBN: 978-1-5386-4633-5, DOI: `10.1109/NETSOFT.2018.8459911`.

[69] R. Ferrús, O. Sallent, J. Pérez-Romero, and R. Agustí, “On 5G Radio Access Network Slicing: Radio Interface Protocol Features and Configuration,” *IEEE Commun. Mag.*, 2018, ISSN: 01636804, DOI: `10.1109/MCOM.2017.1700268`.

[70] J. Perez-Romero, O. Sallent, R. Ferrus, and R. Agusti, “Profit-Based Radio Access Network Slicing for Multi-tenant 5G Networks,” in *2019 Eur. Conf. Networks Commun.*, IEEE, Jun. 2019, pp. 603–608, ISBN: 978-1-7281-0546-8, DOI: `10.1109/EuCNC.2019.8801988`.

[71] V. Sciancalepore, M. Di Renzo, and X. Costa-Perez, “STORNS: Stochastic Radio Access Network Slicing,” in *ICC 2019 - 2019 IEEE Int. Conf. Commun.*, IEEE, May 2019, pp. 1–7, ISBN: 978-1-5386-8088-9, DOI: `10.1109/ICC.2019.8761885`.

[72] A. Papa, M. Klugel, L. Goratti, T. Rasheed, and W. Kellerer, “Optimizing Dynamic RAN Slicing in Programmable 5G Networks,” in *ICC 2019 - 2019 IEEE Int. Conf. Commun.*, IEEE, May 2019, pp. 1–7, ISBN: 978-1-5386-8088-9, DOI: `10.1109/ICC.2019.8761163`.

[73] H. Zhang and V. W. S. Wong, “A Two-Timescale Approach for Network Slicing in C-RAN,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6656–6669, Jun. 2020, ISSN: 0018-9545, DOI: `10.1109/TVT.2020.2985289`.

[74] J. X. Salvat, L. Zanzi, A. Garcia-Saavedra, V. Sciancalepore, and X. Costa-Perez, “Overbooking network slices through yield-driven end-to-end orchestration,” in *Proc. 14th Int. Conf. Emerg. Netw. Exp. Technol.*, New York, NY, USA: ACM, Dec. 2018, pp. 353–365, ISBN: 9781450360807, DOI: `10.1145/3281411.3281435`.

[75] B. Khodapanah, A. Awada, I. Viering, A. N. Barreto, M. Simsek, and G. Fettweis, “Slice Management in Radio Access Network via Iterative Adaptation,” in *ICC 2019 - 2019 IEEE Int. Conf. Commun.*, IEEE, May 2019, pp. 1–7, ISBN: 978-1-5386-8088-9, DOI: `10.1109/ICC.2019.8761376`.

[76] B. Khodapanah, A. Awada, I. Viering, A. N. Barreto, M. Simsek, and G. Fettweis, “Framework for Slice-Aware Radio Resource Management Utilizing Artificial Neural Networks,” *IEEE Access*, vol. 8, pp. 174 972–174 987, 2020, ISSN: 2169-3536, DOI: `10.1109/ACCESS.2020.3026164`.

[77] K. Boutiba, A. Ksentini, B. Brik, Y. Challal, and A. Balla, “NRflex: Enforcing network slicing in 5G New Radio,” *Comput. Commun.*, vol. 181, pp. 284–292, Jan. 2022, ISSN: 01403664, DOI: `10.1016/j.comcom.2021.09.034`.

[78] X. Li, R. Ni, J. Chen, Y. Lyu, Z. Rong, and R. Du, “End-to-End Network Slicing in Radio Access Network, Transport Network and Core Network Domains,” *IEEE Access*, vol. 8, pp. 29 525–29 537, 2020, ISSN: 21693536, DOI: `10.1109/ACCESS.2020.2972105`.

[79] N. A. Mohammedali, T. Kanakis, A. Al-Sherbaz, and M. O. Agyeman, “Performance Evaluation for End-to-End Slice Management in 5G/B5G Cellular Networks,” in *2022 Int. Conf. Software, Telecommun. Comput. Networks*, IEEE, Sep. 2022, pp. 1–6, ISBN: 978-953-290-117-7, DOI: `10.23919/SoftCOM55329.2022.9911525`.

[80] S. Ravindran, S. Chaudhuri, J. Bapat, and D. Das, “Isolation-based Sub-Slices for Throughput Optimization in 5G Radio Access Network,” in *2019 IEEE Int. Conf. Electron. Comput. Commun. Technol.*, IEEE, Jul. 2019, pp. 1–6, ISBN: 978-1-7281-2472-8, DOI: `10.1109/CONECCT47791.2019.9012928`.

[81] S. Ravindran, S. Chaudhuri, J. Bapat, and D. Das, “EESO: Energy Efficient System-resource Optimization of Multi-Sub-Slice-Connected User in 5G RAN,” in *2020 IEEE Int. Conf. Electron. Comput. Commun. Technol.*, IEEE, Jul. 2020, pp. 1–6, ISBN: 978-1-7281-6828-9, DOI: `10.1109/CONECCT50063.2020.9198455`.

[82] S. Ravindran, S. Chaudhuri, J. Bapat, and D. Das, “Efficient Service Allocation Scheduling Algorithms for 5G User Equipments in Slice-in-Slice Networks,” in *2021 IEEE Int. Conf. Adv. Networks Telecommun. Syst.*, IEEE, Dec. 2021, pp. 36–41, ISBN: 978-1-6654-4893-2, DOI: `10.1109/ANTS52808.2021.9937002`.

[83] G. Hasegawa, S. Hasegawa, S. Arakawa, and M. Murata, “UONA: User-Oriented Network slicing Architecture for beyond-5G networks,” in *ICC 2021 - IEEE Int. Conf. Commun.*, IEEE, Jun. 2021, pp. 1–6, ISBN: 978-1-7281-7122-7, DOI: `10.1109/ICC42927.2021.9500653`.

[84] S. K. Singh, M. M. Salim, J. Cha, Y. Pan, and J. H. Park, “Machine Learning-Based Network Sub-Slicing Framework in a Sustainable 5G Environment,” *Sustainability*, vol. 12, no. 15, p. 6250, Aug. 2020, ISSN: 2071-1050, DOI: `10.3390/su12156250`.

[85] C. Y. Chang, N. Nikaein, and T. Spyropoulos, “Radio access network resource slicing for flexible service execution,” *INFOCOM 2018 - IEEE Conf. Comput. Commun. Work.*, pp. 668–673, 2018, DOI: `10.1109/INFCOMW.2018.8407021`.

[86] C. Y. Chang and N. Nikaein, “RAN runtime slicing system for flexible and dynamic service execution environment,” *IEEE Access*, vol. 6, pp. 34 018–34 042, 2018, ISSN: 21693536, DOI: `10.1109/ACCESS.2018.2847610`.

[87] I. Oussakel, P. Owezarski, P. Berthou, and L. Houssin, “Toward Radio Access Network Slicing Enforcement in Multi-cell 5G System,” *J. Netw. Syst. Manag.*, vol. 31, no. 1, p. 8, Jan. 2023, ISSN: 1064-7570, DOI: `10.1007/s10922-022-09694-0`.

[88] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, “OpenAirInterface: Democratizing innovation in the 5G Era,” *Comput. Networks*, vol. 176, p. 107 284, Jul. 2020, ISSN: 13891286, DOI: `10.1016/j.comnet.2020.107284`.

[89] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, “srsLTE: An open-source platform for LTE evolution and experimentation,” *Proc. Annu. Int. Conf. Mob. Comput. Networking, MOBICOM*, vol. 03-07-Octo, pp. 25–32, 2016, DOI: `10.1145/2980159.2980163`.

[90] A. de Javel, J. S. Gomez, P. Martins, J. L. Rougier, and P. Nivaggioli, “Towards a new open-source 5G development framework: an introduction to free5GRAN,” in *2021 IEEE 93rd Veh. Technol. Conf.*, IEEE, Apr. 2021, pp. 1–5, ISBN: 978-1-7281-8964-2, DOI: `10.1109/VTC2021-Spring51267.2021.9448964`.

[91] L. Bonati, M. Polese, S. D’Oro, S. Basagni, and T. Melodia, “Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead,” *Comput. Networks*, vol. 182, p. 107 516, Dec. 2020, ISSN: 13891286, DOI: `10.1016/j.comnet.2020.107516`.

[92] O-RAN, *Open & Intelligent Software for the Radio Access Networks*, 2021, [Online]. Available: `https://www.o-ran.org/software` (visited on 09/23/2023).

[93] Eurecom, *OpenAirInterface5G*, 2021, [Online]. Available: `https://gitlab.eurecom.fr/oai/openairinterface5g` (visited on 09/23/2023).

[94] S. Project, *srsRAN*, 2023, [Online]. Available: `https://www.srsran.com/5g` (visited on 09/23/2023).

[95] Eurecom, *Cn5g*, 2021, [Online]. Available: `https://gitlab.eurecom.fr/oai/cn5g` (visited on 09/23/2023).

[96] *free5GC*, 2022, [Online]. Available: `https://www.free5gc.org/`.

[97] P. K. Gkonis, P. T. Trakadas, and D. I. Kaklamani, "A Comprehensive Study on Simulation Techniques for 5G Networks: State of the Art Results, Analysis, and Future Challenges," *Electronics*, vol. 9, no. 3, p. 468, Mar. 2020, ISSN: 2079-9292, DOI: `10.3390/electronics9030468`.

[98] F. Nizzi, T. Pecorella, M. Bastianini, C. Cerboni, A. Buzzigoli, and A. Fratini, "The Role of Network Simulator in the 5G Experimentation," in *Proc. 2019 Work. Next-Generation Wirel. with ns-3*, New York, NY, USA: ACM, Jun. 2019, pp. 13–17, ISBN: 9781450372787, DOI: `10.1145/3337941.3337949`.

[99] M. K. Müller, F. Ademaj, T. Dittrich, *et al.*, "Flexible multi-node simulation of cellular mobile communications: the Vienna 5G System Level Simulator," *EURASIP J. Wirel. Commun. Netw.*, vol. 2018, no. 1, p. 227, Dec. 2018, ISSN: 1687-1499, DOI: `10.1186/s13638-018-1238-7`.

[100] S. Pratschner, B. Tahir, L. Marijanovic, *et al.*, "Versatile mobile communications simulation: the Vienna 5G Link Level Simulator," *EURASIP J. Wirel. Commun. Netw.*, vol. 2018, no. 1, p. 226, Dec. 2018, ISSN: 1687-1499, DOI: `10.1186/s13638-018-1239-6`.

[101] C.-K. Jao, C.-Y. Wang, T.-Y. Yeh, *et al.*, "WiSE: A System-Level Simulator for 5G Mobile Networks," *IEEE Wirel. Commun.*, vol. 25, no. 2, pp. 4–7, Apr. 2018, ISSN: 1536-1284, DOI: `10.1109/MWC.2018.8352614`.

[102] Y. Kim, J. Bae, J. Lim, *et al.*, "5G K-Simulator: 5G System Simulator for Performance Evaluation," in *2018 IEEE Int. Symp. Dyn. Spectr. Access Networks*, IEEE, Oct. 2018, pp. 1–2, ISBN: 978-1-5386-5191-9, DOI: `10.1109/DySPAN.2018.8610404`.

[103] N. Patriciello, S. Lagen, B. Bojovic, and L. Giupponi, "An E2E simulator for 5G NR networks," *Simul. Model. Pract. Theory*, vol. 96, p. 101 933, Nov. 2019, ISSN: 1569190X, DOI: `10.1016/j.simpat.2019.101933`.

[104] S. Martiradonna, A. Grassi, G. Piro, and G. Boggia, "Understanding the 5G-air-simulator: A tutorial on design criteria, technical components, and reference use cases," *Comput. Networks*, vol. 177, p. 107 314, Aug. 2020, ISSN: 13891286, DOI: `10.1016/j.comnet.2020.107314`.

[105] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5G–An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks," *IEEE Access*, vol. 8, pp. 181 176–181 191, 2020, ISSN: 2169-3536, DOI: `10.1109/ACCESS.2020.3028550`.

[106] A. Dilmaç, M. E. Güre, and T. Tuğcu, *Slicesim: A Simulation Suite for Network Slicing in 5G Networks*, 2020, [Online]. Available: `https://github.com/cerob/slicesim` (visited on 04/21/2021).

[107] CTTC, *5G-LENA Simulator*, 2021, [Online]. Available: `https://5g-lena.cttc.es/` (visited on 09/10/2020).

[108] M. Miozzo, N. Bartzoudis, M. Requena, *et al.*, "SDR and NFV extensions in the ns-3 LTE module for 5G rapid prototyping," in *2018 IEEE Wirel. Commun. Netw. Conf.*, IEEE, Apr. 2018, pp. 1–6, ISBN: 978-1-5386-1734-2, DOI: `10.1109/WCNC.2018.8377237`.

[109] *5G Network Simulator (5G K-SimNet)*, 2019, [Online]. Available: `https://github.com/crazysct/kSimNet` (visited on 02/22/2022).

[110] Telematics, *5G-air-simulator*, 2020, [Online]. Available: `https://github.com/telematics-lab/5G-air-simulator` (visited on 02/03/2021).

[111] 3GPP, "TS 38.214 - NR; Physical layer procedures for data," Tech. Rep., [Online]. Available: `https://www.3gpp.org/DynaReport/38214.htm`.

[112] K. Hooli, P. Kinnunen, E. Tiirola, *et al.*, "Extending 5G to narrow spectrum allocations," *IEEE J. Sel. Areas Commun.*, pp. 1–1, 2023, ISSN: 0733-8716, DOI: `10.1109/JSAC.2023.3273703`.

[113] W. Yang, M. Wang, J. Zhang, *et al.*, "Narrowband Wireless Access for Low-Power Massive Internet of Things: A Bandwidth Perspective," *IEEE Wirel. Commun.*, vol. 24, no. 3, pp. 138–145, Jun. 2017, ISSN: 1536-1284, DOI: `10.1109/MWC.2017.1600298`.

[114] J. MacQueen, "Some Methods for Classification and Analysis of MultiVariate Observations," in *Proc Berkeley Symp. Math. Stat. Probab.*, 1965, pp. 281–297, [Online]. Available: `https://books.google.com/books?hl=en&lr=&id=IC4Ku_7dBFUC&oi=fnd&pg=PA281&ots=nPXeCYIfrN&sig=M-WhsH4-4MIaJZ2gxeNun6UILwc`.

[115] K. P. Sinaga and M.-S. Yang, "Unsupervised K-Means Clustering Algorithm," *IEEE Access*, vol. 8, pp. 80 716–80 727, 2020, ISSN: 2169-3536, DOI: `10.1109/ACCESS.2020.2988796`.

[116] M. I. Malinen and P. Fränti, "Balanced k-means for clustering," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8621 LNCS, pp. 32–41, 2014, ISSN: 16113349, DOI: `10.1007/978-3-662-44415-3_4`.

[117] P. S. Bradley, K. P. Bennett, and A. Demiriz, "Constrained k-means clustering," *Tech. Rep.*, p. 9, 2000, [Online]. Available: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.3257`.

[118] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer (Long. Beach. Calif).*, vol. 36, no. 1, pp. 41–50, Jan. 2003, ISSN: 0018-9162, DOI: `10.1109/MC.2003.1160055`.

Acknowledgements

To begin with, I wish to extend my gratitude to the Thomas Johann Seebeck Department of Electronics at Tallinn University of Technology (Taltech) for providing me with the opportunity to pursue a PhD.

My PhD study was a challenge to me and my supervisors. I would like to express my deepest appreciation to my supervisors for accepting the challenge to supervise me: Dr. Muhammad Mahtab Alam for motivating and leading the process, Dr. Ivo Müürsepp for being available to answer my questions, and Dr. Sven Pärand for giving me a real-life practical viewpoint.

I would like to thank the internal reviewer for the valuable suggestions. The quality of the thesis was significantly improved based on the comments.

I thank all my colleagues and fellow PhD students in our department for their support. Special thanks go to director Laur Lemendik for his encouragement to start and motivation to complete my PhD studies within nominal time. Additionally, I would like to thank Dr. Osama Elgarhy for his help in writing the survey paper, which was rejected a year later.

I would also like to thank Dr. Sven Nõmm from the Department of Software Science for teaching MATLAB and machine learning in the course "Machine learning". This led me to play with MATLAB and machine-learning tools, with a pretty nice outcome.

Thanks should also go to my family for their support during my study, especially the last year.

Finally, I would like to acknowledge the financial support received for this research from various sources, including the European Union’s Horizon 2020 Research and Innovation Program under Grant 951867 ‘5G-ROUTES’ and Grant 101058505 ‘5G-TIMBER’, Tallinn University of Technology Development Program 2016-2022, code 2014-2020.4.01.16-0032, project “ICT programme” supported by the European Union through European Social Fund, and TAR16013 Center of Excellence ‘EXCITE IT’.

Abstract

Radio Access Network Subslicing in Closed Control Loop for Improved Slice Performance Management

The network slicing feature in 5G enables multiple different service level agreements (SLA) to be satisfied using a single physical infrastructure, such that each service vertical or mobile virtual network operator (MVNO) obtains a slice of infrastructure resources. Network slicing enables resource usage efficiency and satisfaction of different SLAs, for example, low latency in one slice and high throughput in another. In a radio access network (RAN), the signal quality of user equipment (UE) determines how much throughput is achieved. Bandwidth allocation methods enable optimization in satisfaction of throughput and delay requirements, and spectral efficiency. Thus, the slice is in a close-to-slice-overload situation. It is not researched, how the size of the bandwidth part (BWP) available to the slice can be used to optimize the slice performance. If there exists the dependence between BWP size and slice performance, subslicing can improve slice performance. Slice performance improvement means that bandwidth utilization has decreased, goodput per one RB has increased, and slice average UE block error ratio (BLER) has decreased.

5G-NR technology is used for the 5G RAN slice. The dependence of slice performance on slice size is investigated by simulating slices of all possible sizes in MATLAB version R2021b. The BWP size is increased by one resource block (RB) which is 180 kHz, if the subcarrier spacing is 15 kHz. For each RB, a UE that can consume the RB is admitted. The UE requested rates in UL and DL are determined by the trials of simulation of slice with maximum BWP size with the aim of achieving close-to-slice-overload situation. The six key performance indicators (KPI) are evaluated: bandwidth utilization, goodput per one RB and BLER, all for both uplink (UL) and downlink (DL). After the discovery of the dependence of slice performance on slice size, subslicing algorithms are proposed. The subslicing algorithm clusters the UEs and subpartitions the slice bandwidth. Bandwidth subpartitions are allocated to UE clusters. For automatic subslicing, a decision mechanism using a classifier neural network (NN) is proposed. The training data are labelled based on solving the optimization problem with the objective of minimizing bandwidth utilization and BLER and maximizing goodput per one RB. Based on subslice performance, NN decides too large subslices to be split, too small subslices to be merged, and suitably sized subslices to be not changed.

The slice simulation results show that slice performance depends on BWP size. The dependence is different; for good-BLER UEs, the best subslice size is greater than for poor-BLER UEs. The decision mechanism determines correctly if it has training samples of both slices that contain all good-BLER UEs or all poor-BLER UEs. Based on the simulation results, subslicing can improve the slice performance if the subslice is not too small in RBs. The proposed subslicing algorithms convert the minimum subslice size constraint to the minimum cluster size constraint for UE clustering. The well-known k-means clustering algorithm was modified to achieve clusters that are not smaller than the minimum cluster size constraint. The subslice bandwidth is allocated proportionally to the number of UEs or the requested sum rate, and the average UE BLER in the cluster.

The simulation results show that the greatest performance improvement by subslicing can be achieved if the UEs are clustered by UE BLER, and larger subslices are created for good-BLER UEs and smaller for poor-BLER UEs. Automatic subslicing in a management closed control loop can improve the slice performance as follows: by selecting the size for a subslice, the subslice bandwidth utilization can be reduced by up to 12%, goodput increases by up to 36% in UL and 11% in DL, or BLER decreases by up to 60%.

Kokkuvõte

Raadiojuurdepääsu võrguviilu alamviilutamine suletud juhtimisahelas

5G võrgu viilutamise funktsioon võimaldab ühe füüsilise taristu kasutamisega pakkuda mitme erineva kvaliteedinõudega teenust nii, et iga teenuste grupp (vertikaal) või virtuaalne võrguoperaator saaks kasutada ainult osa ressurssidest. Võrgu viilutamine võimaldab efektiivset ressursikasutust ja erinevate kvaliteedinõuete rahuldamist, näiteks ühes võrguviilus on lühikest viidet ja teises suurt edastuskiirust vajavad teenused. Olemasolevad sagedusriba jaotamise meetodid võimaldavad optimeerida spektraalefektiivsust ning tagada läbilaskevõime ja viite nõuete täitmine. Raadiovõrgu viilule antud sagedusriba laiuse mõju viilu jõudluse parendamiseks pole uuritud. Kui leitakse viilu sagedusriba laiuse ja viilu jõudluse vahel seos, siis saab viilu jõudluse parandamiseks kasutada alamviilutamist. Viilu jõudluse parandamine tähendab, et sagedusriba kasutus ja kasutajate keskmine blokivea tõenäosus vähenevad ning saavutatud edastuskiirus suureneb.

Raadiovõrgu viilus kasutatakse 5G raadiojuurdepääsu tehnoloogiat NR (*New Radio*). Viilu jõudluse sõltuvus viilu suurusest tehakse kindlaks nii, et simuleeritakse MATLABis (versioon R2021b) raadiovõrgu viil kõigis tema võimalikes suurustes ühe ressursibloki (RB) kaupa. Iga viilule määratud RB-d kasutab üks kasutaja, mis vajab sellist edastuskiirust üleslülis ja allalülis, mis suudab selle RB ära kasutada. Kasutajale vajalikud edastuskiirused määrati katseliselt, kui raadiovõrguviilu suurus on maksimaalne ja viilu sagedusriba kasutuskoormus lähedane maksimaalsele. Pärast seda pakutakse välja alamviilutamise algoritmid. Alamviilutamise algoritm jagab kasutajad gruppidesse ja tükeldab viilule antud sagedusriba. Iga kasutajate grupp saab kasutusse ühe sagedusriba tüki viilu sagedusribast. Automaatse alamviilutamise võimaldamiseks kasutatakse suletud juhtimisahelat. Kasutades eeltreenitud tehisnärvivõrku tehakse iga alamviilu kohta otsus alamviilu jõudluse andmete põhjal, kas alamviil poolitada, liita teise alamviiluga või alamviilu seadistust pole vaja muuta. Treeningandmed märgistatakse optimeerimisülesande lahenduse põhjal, mille eesmärk on vähendada sagedusriba kasutuskoormust ja blokivea tõenäosust ning suurendada saavutatud läbilaskevõimet ühe RB kohta. Tehisnärvivõrgu sisendiks on alamviilu 6 peamist jõudluse näitajat (sagedusriba kasutuskoormus, saavutatud edastuskiirus ühe RB kohta ja blokivea tõenäosus, kõik nii üleslülis ja allalülis) ja väljundiks on otsused.

Võrguviilu simulatsioonitulemused näitavad viilu jõudluse sõltuvust viilule antud sagedusriba laiusest. Kui kasutajate blokivea tõenäosus on väike (hea), siis parima jõudlusega alamviil on suurem, ja kui kasutajate blokivea tõenaosus on suur (halb), siis parima jõudlusega alamviil on väiksem. Otsustusmehhanism teeb õigeid otsuseid, kui treeningandmetes on viilu jõudluse andmeid kõikide väikese blokivea tõenäosusega kasutajate viiludest ja kõikide suure blokivea tõenäosusega kasutajate viiludest. Töös väljapakutud alamviilutamise algoritmid teisendavad alamviilu suuruse RB-dest minimaalse kasutajate klastri suuruse piiranguks, mida kasutab muudetud klasterdamise algoritm. Alamviilule jaotatakse sagedusriba laius, mille suurus on võrdeline kasutajate arvuga või alamviilu summaarse vajaliku edastuskiirusega ning võrdeline kasutajagrupi blokivea tõenäosusega.

Kasutajad tuleks grupeerida nende blokivea tõenäosuse järgi, siis luuakse suuremad alamviilud väikese blokivea tõenaosusega kasutajagruppidele ja väiksemad alamviilud suurema blokivea tõenäosusega kasutajagruppiele. Selline meetod on näidanud kõige suuremat viilu jõudluse paranemist alamviilutamisega. Automaatne alamviilutamine suletud juhtimisahelas võib parandada raadiovõrgu viilu jõudlust järgmiselt: sobiva suurusega alamviiludega võib sagedusriba koormust vähendada kuni 12%, saavutatud läbilaskevõimet suurendada kuni 36% üleslülis ja 11% allalülis või blokivea tõenäosust parandada kuni 60%.

Appendix 1

I

M. Kulmar, I. Muursepp, and M. M. Alam, “The Impact of RAN Slice Bandwidth Subpartitioning on Slice Performance”, in *2022 Int. Wirel. Commun. Mob. Comput.*, IEEE, May 2022, pp. 419–424, ISBN: 978-1-6654-6749-0, DOI: `10.1109/IWCMC55113.2022.9825262`

The Impact of RAN Slice Bandwidth Subpartitioning on Slice Performance

Marika Kulmar
Thomas Johann Seebeck Department of Electronics
Tallinn University of Technology
Tallinn, Estonia
marika.kulmar@taltech.ee

Ivo Müürsepp
Thomas Johann Seebeck Department of Electronics
Tallinn University of Technology
Tallinn, Estonia
ivo.muursepp@ttu.ee

Muhammad Mahtab Alam
Thomas Johann Seebeck Department of Electronics
Tallinn University of Technology
Tallinn, Estonia
muhammad.alam@taltech.ee

***Abstract*—Network slicing in 5G RAN enables building logical networks on top of physical infrastructure. In case of RAN spectrum resources are divided into small bandwidth parts then, the network function in core network, namely Network Slice Admission Control Function (NSACF) enables to limit the count of user equipments (UE) and packet data unit (PDU) sessions in a slice. This paper presents negative results and a practical analysis of how many UEs and PDU sessions can be allowed to use if available bandwidth is fixed and tries to evaluate the optimal count of subslices at this bandwidth. In case of slice resource overutilization there are 2 options: increase resources or move UEs to another slice. In this paper, the feasibility of the other option is evaluated in case of fixed overall bandwidth where resource increase is not possible. Results show that infrastructure can afford as few slices as possible and slicing does not help if theoretical capacity is exceeded.**

***Index Terms*—5G, RAN, slicing, subpartitioning**

I. INTRODUCTION

The frequency resource is a scarce resource essential for radio transmission. This paper contributes on evaluation of the impact of slice bandwidth subpartitioning on slice performance in case of a fixed frequency bandwidth resource has been allocated to the slice.

If one slice is overloaded with traffic then the options are to increase resources to be allocated to the slice or create a new slice. The performance is measured via resource utilization. There are different approaches in the literature about resource allocation to the slice. In [1] it is recommended to measure the utilization of each resource that the slice uses and then determine the next slice lifecycle management (LCM) operation (e.g. slice scaling, new slice creation, slice deletion).

Monitoring count of UEs can be easier than monitoring the resource utilization. The generic slice template (GST) [2] has attributes to provide allowed maximum number of Packet Data Unit (PDU) sessions, and allowed maximum number of UEs. These are scalability attributes and are optional. 3GPP has defined a new function in core - Network Slice Admission Control Function (NSACF) in Rel-17 [3]. NSACF provides slice status reports and notifications for each slice maximum count of UEs per slice and maximum count of PDU sessions per slice. NSACF maintains list of UEs, count of UEs and count of PDU sessions in slice.

The known problem in ultra-dense networks with Internet of Things (IoT) UEs is Radio Access Network (RAN) congestion problem with massive number of small packets [4]. The massive count of UEs cause extra load on resource allocation and massive small packets need finer granularity of resources to be allocated. The solutions include grouping UEs and allocate smaller resources to UE groups - to use the same timeslots, to associate UEs with the suitable BS-s. According to [4] it is more efficient to group UEs into smaller slices to increase throughput and reduce resource utilization.

The achieved throughput is a parameter that has been evaluated to compare the performance of un-sliced network and sliced network. Some different findings exist with relation to throughput and count of slices. In [5] the optimal numerology and optimal scheduler for each slice (generally a throughput-oriented or latency-constrained) are proposed, and they discovered that the un-sliced network served up to 20% more traffic than throughput-oriented slice and latency-constrained slice in combination while using their optimal schedulers and optimal numerologies. Contrary, in [6], the achieved sum-rate can be higher in sliced network than in un-sliced network. Their slice admission with non-overlapping frequency bandwidth and transmit power assignment has achieved up to 120% of throughput compared to un-sliced network. Similarly, in [7] it is shown that the more slices present, the less total bandwidth is used.

In terms of performance, both results have been obtained - higher throughput can be achieved either with or without slicing.

The contributions of this paper are that the slice performance is evaluated by bandwidth utilization and achieved throughput and goodput (effective throughput). The slice with fixed bandwidth is considered and the impact of slice subpartitioning to slice performance is evaluated. The slice bandwidth

Authors' version of the paper accepted in 2022 International Wireless Communications and Mobile Computing (IWCMC 2022). DOI: 10.1109/IWCMC55113.2022.9825262 ©2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

is subpartitioned into subslices and UEs are grouped into subslices. The performance of each subslice is measured and results are combined to evaluate the slice performance.

II. RAN SUBSLICES PROVISIONING

A. Slice life cycle management and closed control loop

This paper uses inductive (test first and then make conclusions) method to evaluate the best count of slices for using specified bandwidth to serve given number of UEs with their requested datarate.

For slice management the 3GPP management system has defined the closed control loop (CCL) management system in [8]. The CCL system contains monitor, analyze, decide, and execute blocks. These blocks are used for automatic management of working slice instances and are shown in Fig. 1. The monitor block collects data about overall bandwidth utilization and each slice bandwidth utilization. Analyze block detects overload and underload. Decide block decides what slice to create, modify or delete according to information about overall utilization and utilization about each slice. Execution block executes the decisions about slices.

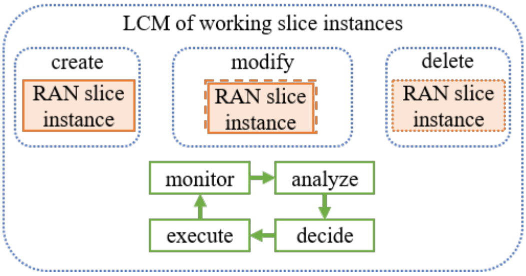


Fig. 1. Slice LCM with closed control loop.

The rules for analyze block are:

- if bandwidth utilization is above 80% then the slice is overloaded, and
- if bandwidth utilization is below 20% then the slice is underloaded

as recommended in [1].

It is recommended to assign UE with the required resources to the slice i.e. apply slice provisioning [9].

The slice has been given a fixed slice bandwidth. The maximum capacity is calculated for this bandwidth. Different number of UEs are planned to use this capacity in the simulations. The more UEs present, the less per-UE data rate is requested. The sum of all per-UE datarates matches the planned percentage of capacity usage. The slice performance is evaluated in case the slice bandwidth is subpartitioned into 1 to N subslices. The UEs with their respective uplink (UL) data rate and downlink (DL) data rate are grouped into subslices. The sizes of UE groups are as equal as possible and UEs are grouped randomly.

B. System model

The slice is a fixed bandwidth resource. The set of UEs are defined with per-UE data rates for UL and DL and UE position coordinates x, y and z are selected randomly within expected coverage area. BS coordinates are $(0, 0, 0)$. The slice resource can be subpartitioned into subslices. The bandwidth for a subslice is calculated from count of subslices for the iteration and count of UEs admitted to a subslice. The bandwidth allocation scheme for subslices is shown in Fig. 2.

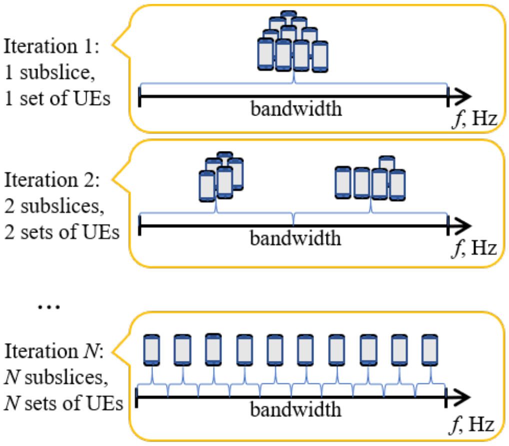


Fig. 2. Bandwidth allocation scheme for subslices at each iteration.

The traffic model is on-off traffic model with the same per-UE data rate for UL and DL. From the specified UE data rate in kbps, the sufficient count of 1500-byte long packets are generated per second. All UEs start transmitting and receiving at the same time.

The scheduler Round Robin was selected to use because this simulation does not measure the efficiency of scheduler but the impact of count of subslices to the slice performance.

The simulation time selected is 100 frames. This is 1 second with 15 kHz subcarrier spacing (SCS) and the snapshot of traffic at time t is simulated.

Other simulation parameters used are shown in table I.

TABLE I
OTHER SIMULATION SETTINGS

Parameter	**Value**
Carrier frequency	3 GHz
Channel model (for both UL and DL)	CDL-C
PUSCH preparation time for UEs	200 μs
Subcarrier spacing	15 kHz
Logical channels per UE	1
RLC entity type	UM bidirectional
Sheduler strategy	Round Robin
Length of scheduling cycle	1 frame
Default packet size	1500 bytes
RB allocation limit UL	same as RBs for subslice
RB allocatin limit DL	same as RBs for subslice

The simulation algorithm divides the slice bandwidth and groups the UEs as equally as possible into given count of subslices starting from 1. This is the benchmark of unsliced situation. Next there are 2 subslices and UEs and bandwidth are divided into 2 subslices. The bandwidth for a subslice is calculated from assigned count of RBs that is calculated from count of UEs assigned to the subslice. The maximum count of subslices to be tested is 10. The simulation algorithm is presented as algorithm 1.

Algorithm 1 Simulation process

```
Set slice bandwidth
Set count of UEs
Set $N \leftarrow$ max count of subslices
for iteration $n = 1$ to $N$ do
    divide UEs and slice bandwidth into $n$ subslices
    for $i = 1$ to $n$ do
        simulate subslice $i$ in Matlab
        collect simulation results of subslice
    end for
    combine simulation results of iteration
end for
plot graphs
```

III. PERFORMANCE EVALUATION

A. Numerical results

The throughput capacity depends on bandwidth and signal-to-noise (SNR) ratio according to Shannon's law [10], equation 1:

$$C = B \cdot \log_2(1 + S/N) \tag{1}$$

where C is channel capacity in bps, B is bandwidth in Hz, S is signal power in mW, N is noise power in mW.

The best and worst SNR values considered are -5 dB and +25 dB accordingly (as done in [11]). Then the theoretical channel capacity is between 17.1 Mbps and 359 Mbps accordingly for given bandwidth of 43.2 MHz.

The data rate depends on channel state and quality that is measured based on reference signals. The UE measures SNR and calculates channel quality index (CQI). According to CQI the tables in 3GPP TS 38.214 [12] section 5.1.3.1 for DL and section 6.1.4.1 for UL are used to select the modulation and coding scheme (MCS) that determines how many data bits are transmitted per symbol and consequently the spectral efficiency.

Now if the spectral efficiency is known, the theoretical capacity can be calculated by equation 2:

$$C = B \cdot \rho \tag{2}$$

where ρ is spectral efficiency in bps/Hz.

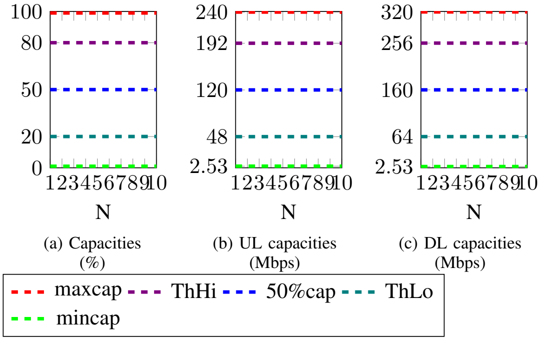
The theoretical maximum capacity of 5G RAN technology NR can be calculated from spectral efficiency found in 3GPP TS 38.214 [12]. In 5G NR the modulations used are QPSK and QAM64 for UL and QPSK, QAM64 and QAM256 for DL. The best spectral efficiency is with modulation of 256QAM and the worst spectral efficiency is with modulation of QPSK.

TABLE II
VALUES FOUND FOR SPECTRAL EFFICIENCY IN 3GPP TS 38.214 [12]

	Spectral efficiency		**Capacity of 43.2 MHz**	
	UL	**DL**	**UL**	**DL**
max	5.5547	7.4063	240 Mbps	320 Mbps
min	0.0586	0.0586	2.53 Mbps	2.53 Mbps

In case the bandwidth is 43.2 MHz and the best and the worst spectral efficiencies are taken from table II left side, then the maximum and minimum capacities can be calculated for UL and DL using equation 2 and results are shown in table II right side.

The percentage of capacities are calculated as percentage of maximum capacity and UL and DL capacities in Mbps are shown in figure 3.



(a) Capacities (%)

(b) UL capacities (Mbps)

(c) DL capacities (Mbps)

Fig. 3. Calculated slice capacities: maximum capacity (maxcap), overload threshold and 80% capacity (ThHi), 50% capacity (50%cap), underload threshold and 20% capacity (ThLo), minimum capacity (mincap).

B. Simulation setup

The subslices are simulated in Matlab R2021b 5G Toolbox system level simulator tool called NR Cell Performance Evaluation with Physical Layer Integration [13].

The minimum bandwidth supported in 5G NR is 4.32 MHz (24 resource blocks (RB) with $SCS = 15$ kHz [14]. The slice bandwidth is planned to divide into a maximum of 10 subslices so for the simulations the slice bandwidth is fixed to 43.2 MHz.

Five different number of UEs are planned to simulate at five different percentages of maximum capacity of a slice. The more UEs the less per-UE data rate is requested such that the sum of per-UE datarates equals to the planned percentage of maximum capacity of a slice.

The maximum number of UEs was selected such that at each slot time at least one RB can be allocated to UE i.e. each UE will have an opportunity to send and receive. The bandwidth of 43.2 MHz in SCS setting 15 kHz is 240 RBs and therefore 240 UEs as a maximum number of UEs are selected to be allowed on the slice bandwidth.

The minimum number of UEs will be 10, because the maximum count of subslices to be tested will be 10 and then each subslice can have at least one UE.

Other number of UEs that will be tested are 192 UEs (80%), 120 UEs (50%) as the middle setting for sake of approximation and 48 UEs (20% of maximum number of UEs). 10 UEs is 4.2% of maximum number of UEs.

Next, the datarate for each UE for UL and DL is calculated in case of theoretical maximum capacity and theoretical minimum capacity. The capacity settings 80% of maximum capacity as overload threshold and 20% of maximum capacity as underload threshold settings are calculated, and for sake of approximation a middle setting 50% of maximum capacity is calculated for each number of UEs setting. The 25 test-cases are calculated and shown in table III.

TABLE III
CALCULATED SETTINGS FOR UEs

Capacity settings	**number of UEs**	**UL capacity and data rate per UE**	**DL capacity and data rate per UE**
max capacity ("red")		**240 Mbps**	**320 Mbps**
max UEs	240	1000 kbps	1333 kbps
80% UEs	192	1250 kbps	1667 kbps
50% UEs	120	2000 kbps	2667 kbps
20% UEs	48	5000 kbps	6667 kbps
min UEs	10	24000 kbps	32000 kbps
80% capacity ("violet")		**192 Mbps**	**256 Mbps**
max UEs	240	800 kbps	1067 kbps
80% UEs	192	1000 kbps	1333 kbps
50% UEs	120	1600 kbps	2133 kbps
20% UEs	48	4000 kbps	5333 kbps
min UEs	10	19200 kbps	25600 kbps
50% capacity ("blue")		**120 Mbps**	**160 Mbps**
max UEs	240	500 kbps	667 kbps
80% UEs	192	625 kbps	833 kbps
50% UEs	120	1000 kbps	1333 kbps
20% UEs	48	2500 kbps	3333 kbps
min UEs	10	12000 kbps	16000 kbps
20% capacity ("teal")		**48 Mbps**	**64 Mbps**
max UEs	240	200 kbps	267 kbps
80% UEs	192	250 kbps	333 kbps
50% UEs	120	400 kbps	533 kbps
20% UEs	48	1000 kbps	1333 kbps
min UEs	10	4800 kbps	6400 kbps
min capacity ("green")		**2.53 Mbps**	**2.53 Mbps**
max UEs	240	10.5 kbps	10.5 kbps
80% UEs	192	13.2 kbps	13.2 kbps
50% UEs	120	21.1 kbps	21.1 kbps
20% UEs	48	52.7 kbps	52.7 kbps
min UEs	10	253 kbps	253 kbps

C. *Results*

The results of bandwidth utilization and achieved throughput and goodput (effective throughput i.e. achieved throughput except retransmissions) are collected at each subslice and combined for all subslices at each iteration. The sllice utilization is average of all subslice utilizations. The slice throughput and goodput is sum of subslice throughputs and goodputs respectively.

1) Bandwidth utilization: The bandwidth utilization is measured by counting RBs that were allocated to UEs divided by all RBs available. The Fig. 4 shows the bandwidth utilization against varying count of subslices that slice has been subpartitioned into. Graphs from (a) to (e) show different number of UEs that use the percentage of planned capacity such that if more UEs then less per-UE data rate requirement and if less UEs then more per-UE data rate requirement.

Simulation results of slice bandwidth utilization show that actual bandwidth utilization is higher than planned. For example if 50% of capacity (50% cap, blue lines) was planned to use then the resource utilization achieved is about 80%. The more UEs present, the higher the bandwidth utilization is at higher capacities.

If maximum UEs (maxUEs) use 43.2 MHz then the best count of subslices could be 4 as there is the lowest utilization. Same for 80% and 50% of UEs on 43.2 MHz. If less UEs then the best count of subslices could be 5. However, the trend lines are not straight but fluctuating. Many packets UEs wanted, were not sent. Small subslices must have certain amount of control plane information to be included for transmission and then UE data has no opportunity to be transmitted. All testcases had more data in buffer at the end of simulation if count of subslices increased.

2) Throughput and goodput: The throughput and goodput of a slice is a sum of throughputs and goodputs achieved in subslices. The Fig. 5 shows UL data in upper row of graphs and DL data in lower row of graphs. Similarly, the left side graphs show the capacity with more UEs and less per-UE data rate and right side graphs show the capacity with less UEs and more per-UE data rate requested. The dashed lines show calculated planned capacities for maximum, 80%, 50%, 20% and minimum utilizations.

Results of throughput and goodput in UL show in Fig. 5 that with more subslices there is less overhead (smaller gap between throughput and goodput). The overhead contains packet retransmissions. The achieved goodput is no higher than calculated maximum capacity in UL. This means that theoretical calculated capacity matches the simulation results.

In DL the achieved goodput does not exceed 80% limit of DL capacity if planned capacity use is maximum or 80% as seen in Fig. 5 second row graphs. The theoretical maxmium capacity was not reached in simulations. The overhead in DL is much smaller than overhead in UL. This means that the DL had less retransmissions and therefore it is more effective.

If more planned capacity used, then with more UEs the throughput is higher than with less UEs. If less planned capacity used then with less UEs the throughput and goodput are higher than with more UEs. (In Fig. 5 graphs the red and violet lines are higher in left side graphs, but teal and green lines are higher on right side graphs.)

3) Summary: Simulation results show that if UEs use planned capacity then bandwidth utilization is higher than planned capacity (as the test results shown using the same color exceed the dashed lines in graphs). The utilization overhead is remarkable. The gap between throughput and goodput decreases, if more subslices are used. The best count of slices from bandwidth utilization is 4 or 5 if more UEs and 5 or 6 if less UEs present at fixed bandwidth of 43.2 MHz. The best count of slices from achieved throughput and goodput is 3 or 4 if more UEs and 4 or 5 if less UEs present at fixed bandwidth of 43.2 MHz.

When looking at all results, then the increase in count of subslices cause the trend to decrease the values of all measured performance metrics.

IV. CONCLUSION

This paper evaluated the impact of slice bandwidth subpartitioning to slice performance on fixed slice bandwidth. Slice performance was evaluated by bandwidth utilization and slice throughput and goodput. The simulation algorithm evaluates the slice performance in case of slice bandwidth subpartitioning into up to 10 subslices. The UEs and bandwidth resource were assigned to subslices proportionally - if UE is admitted to the subslice, then RBs were allocated to that subslice as well i.e. slice provisioning was applied.

In [5] it was observed that un-sliced network performed better. In [6] the achieved rate was higher in sliced network and in [7] less bandwidth was used in sliced network. The results of this paper show that bandwidth utilization depends on number of subslices, UEs and per-UE data rate. Generally, the throughput and goodput are decreasing, if count of subslices is increased.

The count of UEs to be allowed to use the slice is not a constant but depends on the requested per-UE data rate. Better performance is obseved with less UEs with higher per-UE rate requirement than more UEs with lower per-UE rate requirement are using the slice. More UEs with more per-UE rate requirement generate slice overload situation and require more bandwidth. This cannot be solved by bandwidth subpartitioning. Less UEs with less per-UE rate requirement cause bandwidth wastage and this cannot be solved by bandwidth subpartitioning either.

Future work includes simulations with different packet sizes and evaluation of different UE grouping methods for subslices.

ACKNOWLEDGMENT

This work was supported in part from the European Union's Horizon 2020 research and innovation program, within 5GROUTES project, under grant agreement No 951867, and part by the Tallinn University of Technology Development Program 2016-2022, code 2014-2020.4.01.16-0032.

REFERENCES

[1] Slawomir Kuklinski and Lechoslaw Tomaszewski. "Key Performance Indicators for 5G network slicing". In: *2019 IEEE Conf. Netw. Softwarization*. IEEE, June 2019, pp. 464–471. ISBN: 978-1-5386-9376-6. DOI: 10.1109/NETSOFT.2019.8806692.

[2] GSMA. *NG.116 Generic Network Slice Template v6.0*. Tech. rep. 2021, pp. 1–66. URL: https://www.gsma.com/newsroom/resources/ng-116-generic-network-slice-template-v6-0/.

[3] 3GPP. *TS 23.501 System architecture for the 5G System (5GS); Stage 2 (Release 17)*. Tech. rep. 2021. URL: https://www.3gpp.org/DynaReport/23501.htm.

[4] Shree Krishna Sharma and Xianbin Wang. "Toward Massive Machine Type Communications in Ultra-Dense Cellular IoT Networks: Current Issues and Machine Learning-Assisted Solutions". In: *IEEE Commun. Surv. Tutorials* 22.1 (2020), pp. 426–471. ISSN: 1553-877X. DOI: 10.1109/COMST.2019.2916177.

[5] Maria Raftopoulou and Remco Litjens. "Optimisation of Numerology and Packet Scheduling in 5G Networks: To Slice or not to Slice?" In: *2021 IEEE 93rd Veh. Technol. Conf.* IEEE, Apr. 2021, pp. 1–7. ISBN: 978-1-7281-8964-2. DOI: 10.1109/VTC2021-Spring51267.2021.9448814.

[6] Vincenzo Sciancalepore, Marco Di Renzo, and Xavier Costa-Perez. "STORNS: Stochastic Radio Access Network Slicing". In: *ICC 2019 - 2019 IEEE Int. Conf. Commun.* IEEE, May 2019, pp. 1–7. ISBN: 978-1-5386-8088-9. DOI: 10.1109/ICC.2019.8761885.

[7] Yao Sun et al. "Service Provisioning Framework for RAN Slicing: User Admissibility, Slice Association and Bandwidth Allocation". In: *IEEE Trans. Mob. Comput.* (2020), pp. 1–1. ISSN: 1536-1233. DOI: 10.1109/TMC.2020.3000657.

[8] 3GPP. *TS 28.535 Management and orchestration; Management services for communication service assurance; Requirements*. Tech. rep. 2020. URL: https://www.3gpp.org/DynaReport/28535.htm.

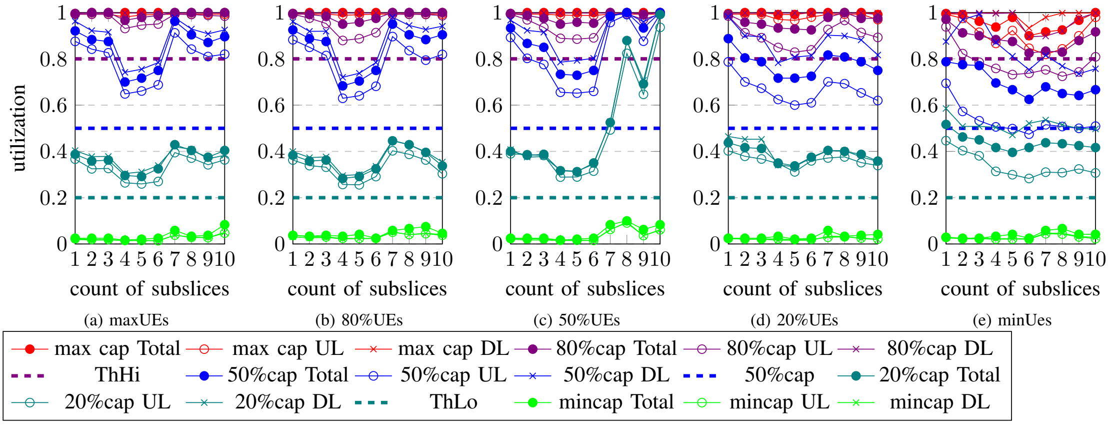
[9] F. Capozzi et al. "Downlink Packet Scheduling in LTE Cellular Networks: Key Design Issues and a Survey". In: *IEEE Commun. Surv. Tutorials* 15.2 (2013), pp. 678–700. ISSN: 1553-877X. DOI: 10.1109/SURV.2012.060912.00100.

[10] C.E. Shannon. "Communication in the Presence of Noise". In: *Proc. IRE* 37.1 (Jan. 1949), pp. 10–21. ISSN: 0096-8390. DOI: 10.1109/JRPROC.1949.232969.

[11] Abitha K Thyagarajan et al. "SNR-CQI Mapping for 5G Downlink Network". In: *2021 IEEE Asia Pacific Conf. Wirel. Mob.* IEEE, Apr. 2021, pp. 173–177. ISBN: 978-1-7281-9475-2. DOI: 10.1109/APWiMob51111.2021.9435258.

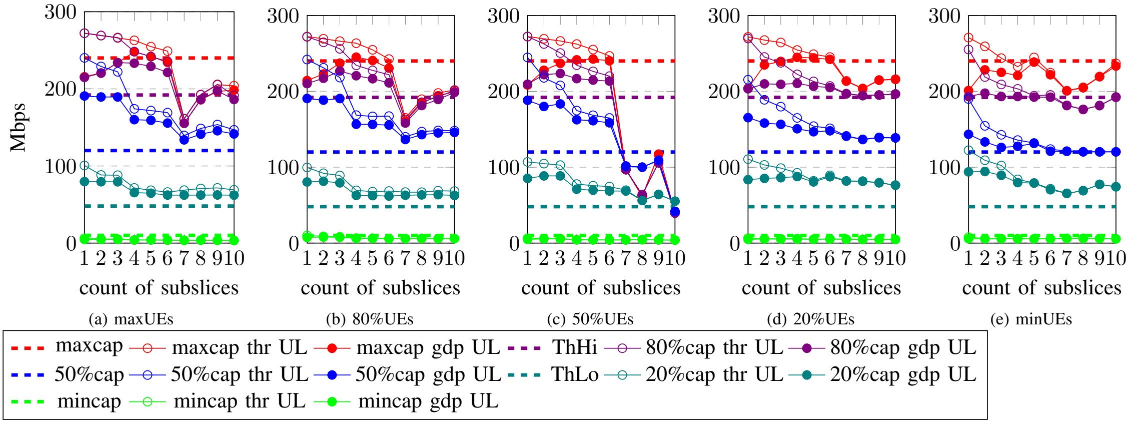
[12] 3GPP. *TS 38.214 V16.5.0 NR; Physical layer procedures for data (Release 16)*. Tech. rep. 2021. URL: https://www.3gpp.org/DynaReport/38214.htm.

[13] MathWorks. *NR Cell Performance Evaluation with Physical Layer Integration*. 2022. URL: https://se.mathworks.com/help/5g/ug/nr-cell-performance-evaluation-with-physical-layer-integration.html (visited on 01/19/2022).

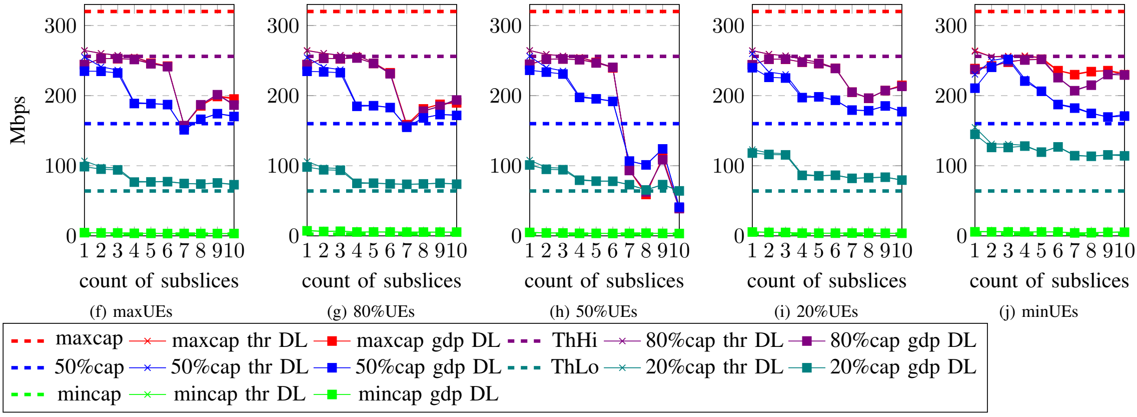


(a) maxUEs (b) 80%UEs (c) 50%UEs (d) 20%UEs (e) minUes

Fig. 4. Simulation results on utilization. Colors: maxcap, 80%cap, 50%cap, 20%cap, mincap.



(a) maxUEs (b) 80%UEs (c) 50%UEs (d) 20%UEs (e) minUEs



(f) maxUEs (g) 80%UEs (h) 50%UEs (i) 20%UEs (j) minUEs

Fig. 5. Simulation results on throughput and goodput on different planned capacities. Colors: maxcap, 80cap, 50cap, 20cap, mincap.

[14] 3GPP. “TS 38.211 NR; Physical channels and modulation”. In: (2020). URL: https://www.3gpp.org/DynaReport/38211.htm.

Appendix 2

II

M. Kulmar, I. Müürsepp, and M. M. Alam, “Heuristic Radio Access Network Subslicing with User Clustering and Bandwidth Subpartitioning”, *Sensors*, vol. 23, no. 10, p. 4613, May 2023, ISSN: 1424-8220, DOI: `10.3390/s23104613`

sensors

MDPI

Article

Heuristic Radio Access Network Subslicing with User Clustering and Bandwidth Subpartitioning

Marika Kulmar *, **Ivo Müürsepp** and **Muhammad Mahtab Alam**

Thomas Johann Seebeck Department of Electronics, Tallinn University of Technology (TalTech), Ehitajate tee 5, 19086 Tallinn, Estonia; ivo.muursepp@taltech.ee (I.M.); muhammad.alam@taltech.ee (M.M.A.)
* Correspondence: marika.kulmar@taltech.ee

Abstract: In 5G and beyond, the network slicing is a crucial feature that ensures the fulfillment of service requirements. Nevertheless, the impact of the number of slices and slice size on the radio access network (RAN) slice performance has not yet been studied. This research is needed to understand the effects of creating subslices on slice resources to serve slice users and how the performance of RAN slices is affected by the number and size of these subslices. A slice is divided into numbers of subslices of different sizes, and the slice performance is evaluated based on the slice bandwidth utilization and slice goodput. A proposed subslicing algorithm is compared with k-means UE clustering and equal UE grouping. The MATLAB simulation results show that subslicing can improve slice performance. If the slice contains all UEs with a good block error ratio (BLER), then a slice performance improvement of up to 37% can be achieved, and it comes more from the decrease in bandwidth utilization than the increase in goodput. If a slice contains UEs with a poor BLER, then the slice performance can be improved by up to 84%, and it comes only from the goodput increase. The most important criterion in subslicing is the minimum subslice size in terms of resource blocks (RB), which is 73 for a slice that contains all good-BLER UEs. If a slice contains UEs with poor BLER, then the subslice can be smaller.

Keywords: 5G; RAN; slicing; UE clustering; performance evaluation

check for updates

Citation: Kulmar, M.; Müürsepp, I.; Alam, M.M. Heuristic Radio Access Network Subslicing with User Clustering and Bandwidth Subpartitioning. *Sensors* **2023**, *23*, 4613. https://doi.org/10.3390/s23104613

Academic Editors: Jehad Ali and Hsiao-Chun Wu

Received: 4 April 2023
Revised: 4 May 2023
Accepted: 7 May 2023
Published: 10 May 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Network slicing in 5G cellular communication network is used to guarantee the service-level agreement (SLA) of a variety of user equipment (UE) instead of providing a best-effort networking service. Slices of network resources as logical separate networks can be created, modified and deleted automatically. The network configuration is dynamically adjusted to serve different groups of UEs. There are no upper bounds on the number of slices and slice size in the network.

Slice performance management is performed in the network slice lifecycle [1] by evaluating the slice performance. If a slice overload occurs, more resources are allocated to the slice. The computing, storage, and networking resources can be mapped to the QoS requirements by the orchestrator. The radio resources are limited and scarce. If additional resources are not available, then a slice overload can be avoided by not serving some UEs; however, then the slice SLA cannot be guaranteed. The mapping of radio resources to the QoS requirements is more complex. With subslicing the slice bandwidth is sub-partitioned into smaller parts to serve smaller UE groups.

From our previous work [2], it is known that the slice performance depends on how many subslices it is divided into. Given that different numbers of subslices affect the slice performance, a suitable number of subslices and subslice sizes exist. A subslicing algorithm can be created that uses found criteria on the subslice size and number of subslices that contribute to the slice performance improvement on a fixed slice bandwidth.

1.1. Related Work

The network slice subnet defined by 3GPP TS 28.530 [1] is a group of network functions that can be managed independently. It is easy to replace the words “network slice subnet instance” (NSSI) with a shorter word, “subslice”. The NSSI is a part of the core, RAN, edge, cloud, etc. network resources that can be separately incorporated into a slice to represent its part of network.

The slice identifier defined in 3GPP TS 23.501 [3] consists of a slice/service type (SST) that denotes standardized verticals or custom numbers and a slice descriptor (SD) that is optional and distinguishes multiple slices with the same SST. Reference [4] defined subslices as slices with the same SST but different SDs and evaluated the end-to-end performance with hundreds of slices in the system. The UE can connect to multiple slices, and slices can be created and deleted. However, RAN resources are assumed to be infinite, and the RAN is simulated using UERANSIM (where the physical layer is not implemented, and the radio interface is simulated over a UDP protocol), which means ideal radio quality and immediate transmission in the RAN is assumed.

Another study [5] proposed a subslicing method where the UE features were selected using a support vector machine (SVM) and the UEs were grouped based on the selected features using k-means. The number of subslices was determined based on the clustering quality measured with the Silhouette coefficient. However, this approach did not consider the performance when creating the subslices, which can lead to a poor performance for small subslices. Their simulations were conducted using Android UEs in Wi-Fi, and the performance of 5G-NR RAT was not evaluated.

On the other hand, Ref. [6] proposed a subslicing method where a subslice in a RAN is treated as a virtual cell that includes multiple physical cells. This approach aims to improve the slice performance by reducing the signaling required for cell handovers within the RAN subslice.

Lastly, a series of studies [7–9] attempted to address subslicing and its impact on performance. In [7], subslices were created for each vertical to include groups of UEs based on their similar SLA values. In [8], the focus was on optimizing the resource allocation for services, where UEs can connect to multiple subslices within a slice. The system load was defined as the range of the number of packets and packet sizes that will not fully utilize the allocated bandwidth. The related work on subslicing is compared in Table 1.

While all studies aimed to improve performance through subslicing, none of them considered the design of subslices that would achieve this goal in the RAN. It is important to note that the simulations used in these studies had a simplified RAN, and the requested rates were too variable to achieve a controlled load that would fully utilize the available bandwidth.

Key performance indicators (KPI) are used to evaluate the slice performance and trigger slice modification. The utilization of each resource as slice run-time KPIs are proposed in [10] to measure the slice performance. If the resource utilization exceeds a given high threshold (e.g., 80 %), then a slice overload can be detected, and slice modification can be triggered to add more resources or drop UEs. Similarly, in [11] high and low thresholds of computing resource utilization are used to trigger slice scaling, which is performed by adding or removing resources. Different values of slice overload thresholds are used. If new resources can be allocated, the slice overload threshold is 80–90%. If no resources are available and resources are taken from another less-loaded slice, the slice overload threshold is 60%. The slice overload threshold is useful for detecting slice overload earlier than it happens to avoid slice malfunction due to insufficient resources.

For subslicing, it is necessary to determine which UE should be served by which subslice. This can be achieved by clustering the UEs. UE clustering has been applied in several studies. In [12] the resource allocation scheme contains the allocation of a resource block (RB) on a time-frequency scale to a cluster of massive machine-type communications (mMTC) UEs in a MIoT slice; however, there are no details about how the UEs are clustered. As for the scheduler, one RB may be sufficient; however, one RB is too small for a subslice as a logical network on physical layer of 5G-NR. In [13] the UEs are clustered using the k-means clustering algorithm into inner-cell UEs and cell-edge UEs using the UE distance from the base station (BS). The bandwidth allocation is different for UE clusters: cell-edge UEs are allocated many smaller-bandwidth parts served by multiple close BSs, whereas inner-cell UEs are allocated bandwidth parts served only by the associated BS. In [14] the UEs are clustered by the UE position to decide how many small-cell BSs are needed to switch on or off to maximize the energy efficiency of the network. The clustering algorithm is based on fuzzy c-means clustering. Similarly, in [15] UEs are clustered by the UE position, and the optimal route for the flying BS is calculated for UE clusters. UE grouping or clustering has rarely been used for network slicing. In [16] a deep neural network is trained to classify UEs into enhanced mobile broadband (eMBB), MIoT, or ultra-reliable low-latency communication (URLLC) slices. In [17], the slice mobility is discussed. The UEs are grouped by the values of parameters that can indicate the UE behavior pattern in the network, that is, related to the association of the moving UE with different BSs. The operations with groups of UEs are related to slice life cycle management (LCM) and group handover, which is faster than triggering and performing the handover of each UE separately.

Table 1. Comparison of related work in subslicing.

Paper	How Subslicing Is Performed	Benefits	Limitations
[7]	Three verticals each contain subgroups of services. UEs grouped by similar SLA values	Performance improvement achieved by subslicing: increased SNR and throughput.	The planned use of capacity varies between underload and half of the full load. The effect of the number of subslices on slice performance has not been evaluated.
[8]	A subslice is a logical group of services that is associated with a single UE. The UE can connect to multiple subslices.	The performance improvement achieved by subslicing: increased throughput and energy efficiency.	The planned use of capacity is variable between underload and close to full load.
[9]	One subslice can serve not only UEs with similar SLAs, but also a mixed set of UEs with different throughputs.	The performance improvement achieved by subslicing: increased throughput.	The planned use of capacity cannot be evaluated using the given data.
[5]	UEs are clustered by the similarity of their requirements.	Subslicing results in decreased bandwidth consumption, improved load balancing, improved latency and heterogeneity, and improved energy efficiency.	Planned underload, RAN simulated as Android UEs in Wi-Fi. The proposed subslicing method does not avoid creating subslices that are too small.
[6]	In the RAN subslice, the virtual cell covers multiple physical cells, and the UE by mobility allocates the virtual cell.	The latency and throughput can be improved because UE handover is performed faster.	Evaluation of just signalling for handover. The UE can select subslices without constraints on the subslice performance.
[4]	The slice with identifier SST contains all slices with the same SST and different SDs as subslices.	Performance improvement was achieved by subslicing.	The RAN was simulated as ideal.
This paper	The slice bandwidth is subpartitioned, slice UEs are grouped, and bandwidth subpartitions are allocated to UE groups. **The number and sizes of subslices are determined with the aim of achieving better performance than without subslicing.**	Slice performance can be improved by reducing bandwidth utilization and increasing goodput if the subslices are not too small. Simulations were performed using **5G-NR and close to the full capacity** of the allocated bandwidth. UE requested rates are sufficient to utilize one RB per UE.	Simulations were performed under the assumptions that all UEs are similar to their requirements and capabilities. The proposed algorithm requires significant computational resources.

After initialization, the original k-means clustering algorithm [18] consists of three steps: distance calculation, cluster assignment, and a new centroid calculation. The algorithm finishes at convergence when the centroids do not move. The cluster size depends on the number of data points that are closer to the specific centroid and not to the other centroids. The cluster size is not limited. It can be empty or contain all data points. Furthermore, there are no outliers as each data point is assigned to a cluster.

K-means has many modifications, ranging from more unsupervised (no number of clusters given) to more constrained and balanced (all clusters of the same size). If an unsupervised k-means algorithm [19] is used, the algorithm determines the number of clusters to be created. This is based on data point similarity; however, we need to specify the desired number of subslices. The constrained k-means [20] algorithm enables the definition of constraints such as the minimum cluster size, and the optimization problem is solved using a linear programming method. Balanced k-means [21] results in clusters of equal size by defining artificial points close to the centroid and using the Hungarian algorithm to pair the data points with artificial centroid points. Agglomerative clustering algorithms begin with small clusters and attempt to merge similar clusters in each step. Finally, some clusters may include only one data point.

The subslicing has been implemented by grouping the services and UEs by their similarities. The effect on performance improvement has been noticed. The state of the art does not consider the following aspects:

- Regarding slice performance evaluation, most studies have considered throughput and/or delay as the main metrics. However, goodput (application-level throughput) should be taken into account, because throughput alone does not provide details about overheads (packet headers and packet retransmissions).
- Slice performance evaluation does not consider resource utilization, which is needed to understand the number and size of subslices to achieve the best performance.

1.2. Motivation and Problem Description

From our previous work, it can be seen that the slice performance depends on the number of subslices. The more subslices there are, the smaller one subslice is. How does the subslice performance depend on the subslice size?

When we see that subslices at some size have a better performance than others, then the slice could be subsliced into subslices of suitable size. What size and how many subslices could be created in the slice to improve the slice performance on a fixed slice bandwidth?

To evaluate how subslicing affects the slice performance, the following research problem is defined:

- **Input**: 275 UEs, 50 MHz (275 RBs), number of subslices in a slice, slice performance data (utilization, throughput, goodput, BLER) when the slice is not subsliced.
- **Decision**: Select the number of subslices to create and select the subslicing method.
- **Objective**: Improve the slice performance (reduce the bandwidth utilization and increase the slice goodput).

1.3. Contributions

The contribution is to answer the above-mentioned research questions and the specific outcome is as follows:

- Present the dependence of subslice performance on the subslice size.
- Propose a subslicing algorithm that prevents creating too-small subslices.
- Compare the slice performance, if it is subsliced using the proposed algorithm, equal UE grouping, or k-means UE clustering algorithms.

The remainder of this paper is organized as follows. In Section 2, the concept of subslicing is described and the performance of a subslice, depending on its size, is evaluated to determine the minimum subslice size. The proposed UE clustering algorithm is described in Section 3. Section 4 contains the slice simulations and slice performance-evaluation results when the slice is subsliced using three different subslicing algorithms. Finally, Section 5 concludes the paper.

2. Subslice Performance Evaluation at Different Subslice Sizes

In this section, the performance of the subslice at different sizes for the selected test case is evaluated.

2.1. Subslicing

Network slicing is a feature of 5G that is applicable to the 5G RAN with a standalone architecture. The RAN slice subnet, or the RAN slice, consists of a gNB, which is implemented as virtual network functions (VNFs) that use computing and storage resources to run and networking resources to enable connectivity between VNFs. The RAN slice also utilizes radio resources (frequency bandwidth) to transfer the UE data and control information using physical network functions, e.g., antennas [1].

To monitor the performance of the operational slices, a management closed control loop is utilized [22], which decides when the slice needs to be modified, and in case of slice overload, more resources can be added during the slice-modification phase [1]. This paper proposes subslicing as a means of slice modification to address a slice overload at a fixed slice bandwidth.

Subslicing is a technique where the original slice bandwidth is divided into smaller parts. The slice UEs are grouped into smaller groups, and these smaller BWPs are allocated to these smaller UE groups. Figure 1 illustrates this approach. It is important to note that all subslices of a slice use slice VNFs and other slice resources.

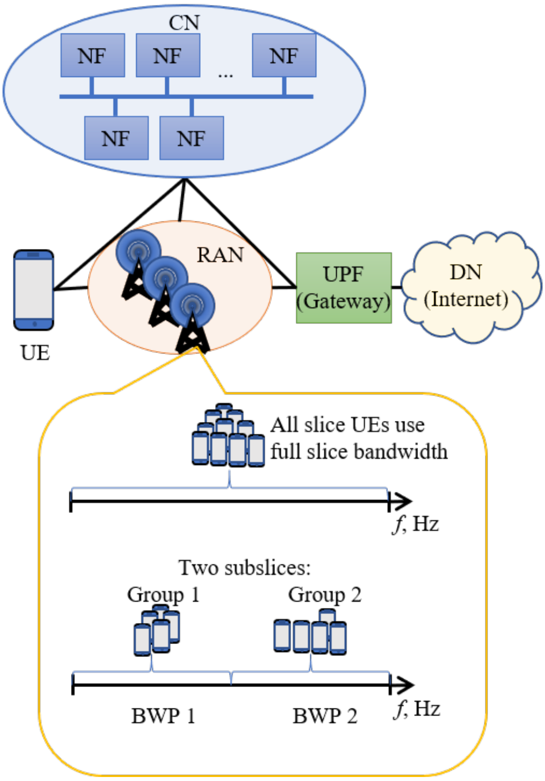


Figure 1. RAN architecture with slicing and subslicing.

To address a slice overload when there is insufficient bandwidth available, subslicing can be applied. The benefit of subslicing is that it can reduce the slice bandwidth utilization while simultaneously increasing the slice goodput, without requiring additional bandwidth allocation, thus ensuring rate requirements and serving more UEs.

The life cycle of a working slice instance and the proposed RAN subslicing are illustrated in Figure 2. Similar subslicing has been carried out in [23], where dynamic inter-slice radio resource partitioning in the time-frequency plane is proposed. The optimization goal is to find the largest unallocated space. The bandwidth parts of the slices can be placed freely inside the time-frequency plane of the infrastructure radio resources.

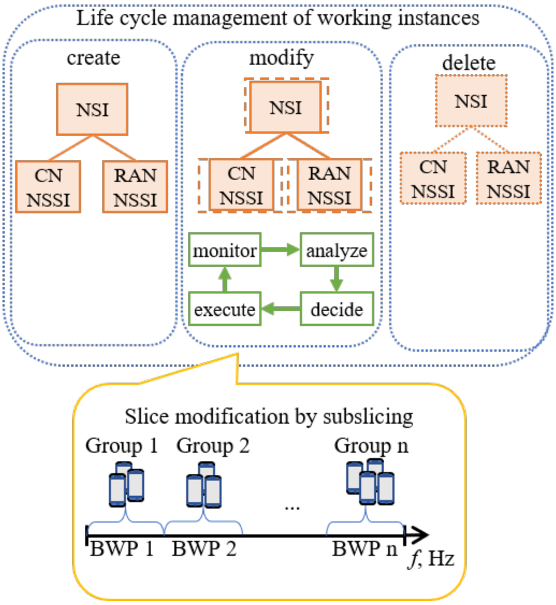


Figure 2. The subslice consists of a subset of slice UEs and a fraction of the slice bandwidth.

The slice resource placement on the time-frequency plane is performed by inter-slice schedulers, but our subslicing framework resides on top of schedulers to subpartition the fixed bandwidth allocated to a slice to improve the slice performance.

2.2. Subslice Simulation Setup

The aim of this simulation is to study the dependence of subslice performance on the subslice size. The one-second working time of the subslice is simulated using the MATLAB 5G toolbox tool “NR Cell Performance Evaluation with Physical Layer Integration” [24]. The subslice has one RB of bandwidth resources allocated per UE in a subslice. In the simulations, the number of RBs and UEs in a subslice starts at four and increases by one until 275. All UEs are similar. The UE request rates are 500 kbps in the UL and 667 kbps in the DL. With these rates, it is possible to achieve approximately 80% utilization, and it allows seeing both an increase and decrease in utilization when subslicing. Each UE is positioned within 174 m (each of the three coordinates within 100 m) from the gNB and is expected to achieve a good BLER below 0.1. A value of 1500 bytes is the default value of the maximum supported packet size if the packet size is not specified in the slice template and 40 bytes is a short packet suitable for the MIoT slice [25]. The subslice simulation settings are presented in Table 2.

Table 2. Subslice data.

Parameter	**Value**	
Number of UEs in a subslice	{4–275}	
Number of RBs allocated to subslice	{4–275}	
Subslice modification for next step	Increase UEs by one and increase RBs by one	
Subcarrier spacing	15 kHz	
UE rate requirements	UL 500 kbps, DL 667 kbps	
	Subband size	
Size of BWP in RBs	**Subband size specified in TS 38.214**	**Subband size used in simulations**
4–23	-	4
24–72	4, 8	8
73–144	8,16	8
145–275	16, 32	16

One parameter is the subband size, which depends on the allocated bandwidth part (BWP) in the RBs. The values of the subband size are defined in 3GPP TS 38.214 [26]. The subband size is used in the channel state information reporting. The other simulation parameters used in MATLAB are listed in Table 3. The subslice performance is measured by means of bandwidth utilization, subslice throughput (thr), and goodput (gdp) for UL and DL, and BLER for UL and DL.

Table 3. MATLAB Toolbox settings.

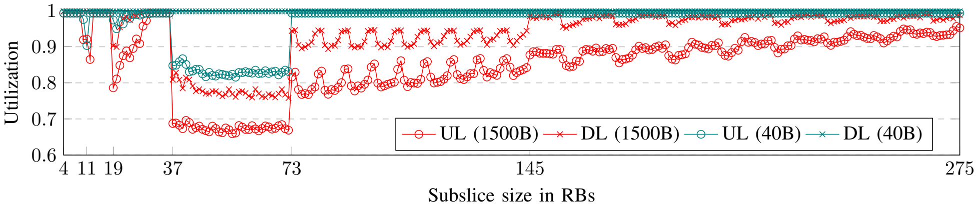
Parameter	**Value**
Carrier frequency	3 GHz
Channel model (for both UL and DL)	CDL-C
PUSCH preparation time for UEs	200 µs
Logical channels per UE	1
RLC entity type	UM bidirectional
Duplex mode	FDD
Scheduler strategy	Round Robin
Length of scheduling cycle	1 frame
RB allocation limit UL	same as RBs for subslice
RB allocation limit DL	same as RBs for subslice
Simulation time	1 s
Subslice simulation tool from MATLAB 5G Toolbox	NR Cell Performance Evaluation with Physical Layer Integration [24] R2021b

2.3. Subslice Simulation Results

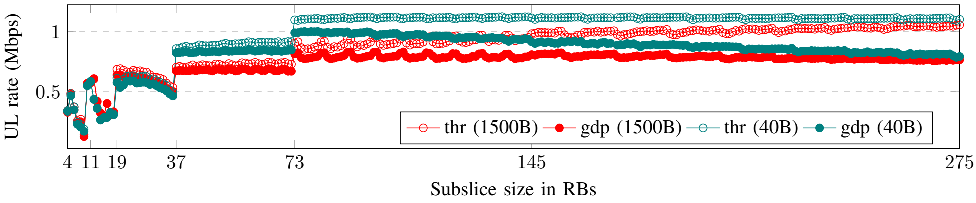
The subslice performance evaluation results, depending on the subslice size, are shown in Figure 3. The subslice bandwidth utilization, throughput, and goodput per RB and the average BLER are collected in both UL and DL. Regarding the packet size, longer packets result in lower bandwidth utilization, whereas shorter packets result in higher throughput and goodput. The BLER does not depend on the packet size, because the block size does not depend on the packet size.

There are four different ranges of subslice sizes, called zones, in which the subslice performance is similar. The performance zones are shown in Figure 4 and the averages of the zone performance data are listed in Table 4.

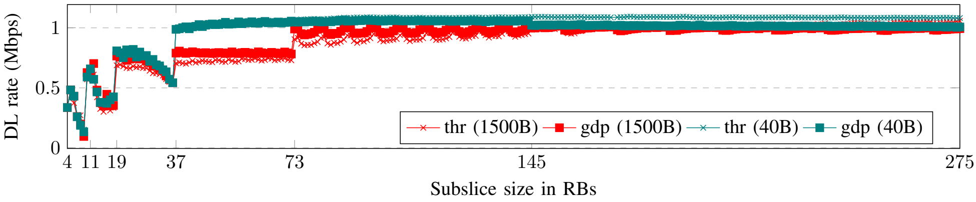
The graph in Figure 3a displays the subslice performance in terms of the bandwidth utilization. The utilization is higher in DL than in UL, and with short packets, the utilization is higher than with long packets. For small subslices with sizes between 4 and 36 RBs (Zone 1), the bandwidth utilization is high. When the subslice size is between 37 and 72 RBs (Zone 2), the utilization drops to its lowest point. The utilization sharply increases to the slice overload threshold when the subslice size is between 73 and 144 RBs (Zone 3). Finally, when the subslice size is greater than 145 RBs (Zone 4), there is a significant boost in utilization.



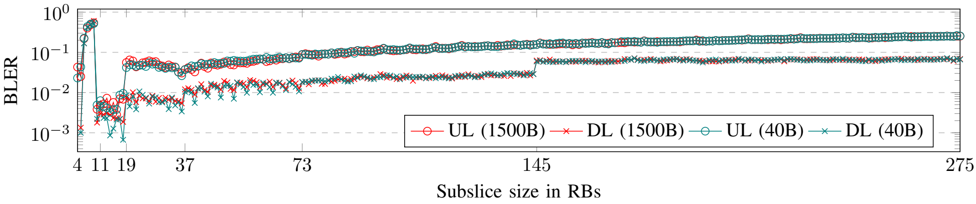
(**a**) Bandwidth utilization



(**b**) Throughput and goodput per RB in UL



(**c**) Throughput and goodput per RB in DL



(**d**) Subslice BLER

Figure 3. Subslice performance depending on subslice size in RBs. (**a**) Subslice bandwidth utilization, (**b**) UL and (**c**) DL throughput (thr) and goodput (gdp) per RB, (**d**) subslice BLER.

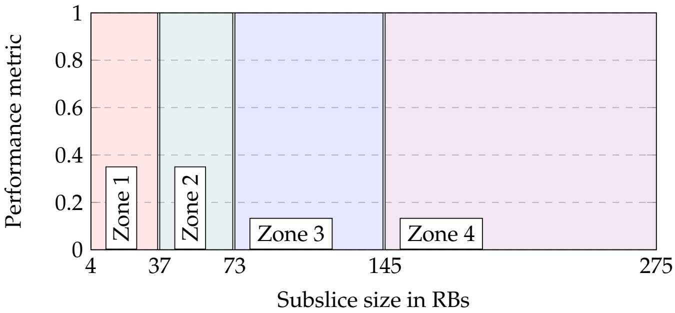


Figure 4. Performance zones depending on subslice size.

Table 4. Average subslice simulation results for subslice size ranges. Best values of each parameter are shown in bold.

Zone	Subslice Size (RBs)	Average Utilization				Average Throughput				Average Goodput				Average BLER			
		1500 B		40 B		1500 B		40 B		1500 B		40 B		1500 B		40 B	
		UL	DL	UL	DL	UL	DL	UL	DL	UL	DL	UL	DL	UL	DL	UL	DL
1	4–36	0.949	0.988	0.986	0.998	0.522	0.571	0.485	0.586	0.493	0.566	0.457	0.581	0.081	0.056	0.079	0.053
2	37–72	**0.675**	**0.777**	**0.831**	0.998	0.726	0.804	0.896	1.044	0.68	0.791	0.838	1.031	**0.057**	**0.015**	**0.061**	**0.013**
3	73–144	0.815	0.923	0.993	0.998	0.915	1.000	**1.115**	1.082	**0.796**	0.975	**0.97**	**1.057**	0.122	0.024	0.122	0.024
4	145–275	0.905	0.978	0.993	0.998	**1.014**	**1.065**	1.113	**1.087**	0.786	**0.994**	0.863	1.015	0.205	0.064	0.206	0.064

Figure 3b,c illustrate the throughput (thr) and goodput (gdp) per RB for different packet sizes in UL and DL, respectively. The simulation results indicate that DL achieves a slightly higher throughput and goodput than UL. Additionally, the same requested data rate with short packets results in a higher goodput. The gap between the throughput and goodput contains the retransmission overhead, which is larger in UL than in DL. The small subslices in Zone 1 have a low throughput and goodput. As the subslice size increases to Zone 2, the throughput and goodput increase to a steady level, with short packets having higher throughput and goodput. In Zone 3, the throughput and goodput increase further, except for DL and short packets. In Zone 4 the goodput decreases in the UL.

The subslice BLER is shown in Figure 3d. Between subslice sizes of 19 and 37 RBs, the BLER gradually decreases before increasing slowly with increasing subslice size. The UL BLER is higher than that of the DL BLER. For the DL, the BLER is below 0.1 for subslice sizes in Zones 2–4. The UL BLER is also below 0.1 in Zones 2 and 3, but in Zone 4, the BLER increases gradually.

The zone boundaries refer to the BWP sizes where the value of a performance metric changes abruptly. This boundary coincides with the size of the RB group (RBG) changes, as specified in Configuration 1 in 3GPP TS 38.214 [26]. The Round Robin scheduler allocates RBGs for each UE for a slot time. The modulation and coding scheme (MCS) and code rate are selected based on the reported channel quality indicator (CQI). The transport block size (TBS) is calculated as specified in Sections 5.1.3.2 (DL) and 6.1.3.2 (UL) of 3GPP TS 38.214. The UEs in slices with smaller BWPs achieve a lower CQI. This is because a smaller BWP has fewer reference symbols available for channel estimation, which can lead to incorrect channel estimation and inappropriate TBS selection. If the channel is estimated to be better than its actual value, then a smaller TBS is selected, resulting in a lower achieved rate. If the channel is worse than the estimated value, a block error occurs. The performance pattern dependent on the BWP size can be repeated if the average MCS of the UEs is calculated for each BWP size, excluding the MCSs for the first three slots when the channel state information (CSI) is unavailable.

The Zone 1 subslice shows poor performance due to its high utilization and low goodput. Zone 2 has the lowest utilization and BLER. Zone 3 has a high goodput with short packets, and Zone 4 has a high goodput with long packets, but a high BLER in UL with both packet sizes. The selected minimum subslice size values are 37 and 73 RBs for the proposed subslicing algorithm, respectively. The former has better utilization, whereas the latter has better goodput.

Ref. [27] conducted a measurement campaign on a 5G-NR gNB using n78 (60 MHz TDD 4:1) with 4 × 4 MIMO for DL and 2 × 2 MIMO for UL. The results showed that a comparable result achieved without MIMO per RB is 0.8 Mbps for DL and 0.9 Mbps for UL. According to our simulation results, a slice with a size of 275 RBs (50 MHz) can achieve a throughput of around 1.1 Mbps per RB and a goodput of 0.8 Mbps per RB in UL and 1 Mbps per RB in DL.

3. Proposed User Clustering and Bandwidth-Allocation Algorithms for Enhanced Slice Performance

In this section, the proposed UE clustering with a bandwidth-allocation algorithm for subslicing is presented. The subslice performance results show that subslices that are too small will degrade the overall slice performance. The proposed clustering algorithm avoids creating clusters of UEs that are too small to allocate too few RBs for a subslice. In addition, a bandwidth-allocation algorithm for UE groups to allocate RBs proportional to the UEs in a group and a group BLER is proposed.

3.1. System Model

The slice is described as a bandwidth resource, a BWP in RBs, $N^{(RB)}$ and number of UEs, $N^{(UE)}$. The minimum subslice size requirement is denoted by S_{min}. The number of subslices requested is K.

The slice UEs are clustered for subslices. Each subslice k contains a subset (group) of slice UEs such that $N^{(UE)} = \sum_{k=1}^{K} N_k^{(UE)}$ and allocates a number of slice RBs such that $N^{(RB)} = \sum_{k=1}^{K} N_k^{(RB)}$ holds. All UEs are assumed to request the same rates. The given number of subslices and minimum subslice size constraint are assumed feasible. That is, with a given minimum subslice size constraint it is possible to create at least the requested number of minimum size or larger subslices, $K \leq \frac{N^{(RB)}}{S_{min}}$.

For clustering UEs into smaller groups for subslices the minimum cluster size can be calculated from the slice RBs and slice UEs:

$$m_{min} = \left\lceil S_{min} \cdot \frac{N^{(UE)}}{N^{(RB)}} \right\rceil. \tag{1}$$

Then UEs are clustered into K clusters with minimum cluster size m_{min}.

The slice bandwidth $N^{(RB)}$ is allocated to UE clusters $N_k^{(UE)}$ in three steps: initially proportional to the number of UEs, secondly proportional to the number of UEs in a group and group BLER, and finally to subslices that are still too small.

The group BLER for UEs $\mathbf{u}_i$ belonging to a cluster C_k is calculated as the average of UE BLERs:

$$BLER_{C_k} = \frac{\sum\limits_{\mathbf{u}_i \in C_k} BLER_{\mathbf{u}_i}}{N_k^{(UE)}}. \tag{2}$$

3.2. Proposed UE Clustering Algorithm

The proposed algorithm is based on k-means clustering. The principle of the algorithm is illustrated using the example shown in Figure 5. In this example, the data points cluster well into two clusters, but it is necessary to cluster them into three clusters with a minimum cluster size of two. The proposed clustering algorithm is described next. In the first assignment step, each cluster takes the required number of data points closest to the cluster centroid. In the second assignment step, the unassigned data points are assigned to the cluster with the closest centroid (connections with arrows).

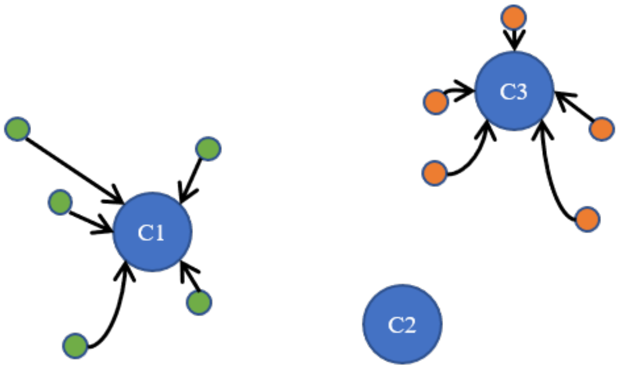
Let $\mathbf{u}_i$ be the value of the metric used to cluster UEs, here UE BLER of the ith UE, $i = 1, \ldots, N$ and $\mathbf{c}_k$ be the centroid of cluster C_k, $k = 1, \ldots, K$. The distances are calculated using the Euclidean distance formula, (3),

$$d_{k,i} = \sqrt{(\mathbf{u}_i - \mathbf{c}_k)^2}. \tag{3}$$

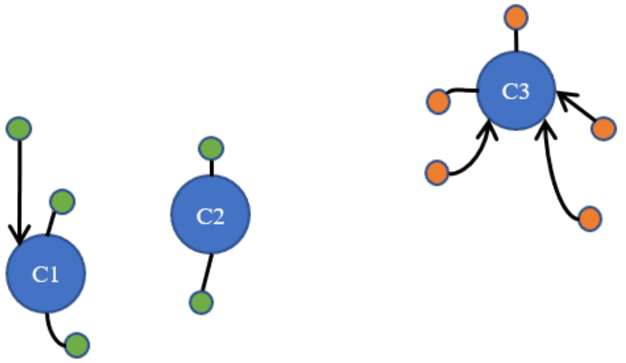
The distance matrix (4) is specified as follows:

$$\mathbf{D} = ||\mathbf{d}_{k,i}||_{n \times N^{(UE)}}. \tag{4}$$

where $d_{k,i}$ is the distance between UE i and the centroid of cluster k.



(a) Original k-means clustering into 3 clusters, cluster 2 is empty



(b) Proposed algorithm clustering into 3 clusters with minimum cluster size of 2 points

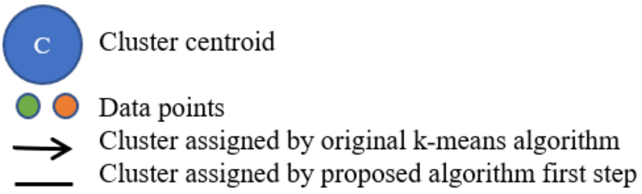

C
Cluster centroid
Data points
Cluster assigned by original k-means algorithm
Cluster assigned by proposed algorithm first step

Figure 5. An example of how the proposed clustering algorithm clusters data points into three clusters with a minimum cluster size of 2. The data points cluster well into two clusters (green and orange).

The original k-means algorithm [18] goes through the columns of the distance matrix and finds the minimum value of the column, and the row index designates the cluster to be assigned to the UE. This algorithm can leave some clusters empty in the worst-case scenario. To avoid empty clusters, the proposed algorithm goes through the rows and finds the minimum of each row for the column vector $\mathbf{M}$ and the index of UE, which has the minimum distance.

$$\mathbf{M} = ||\mathbf{m}_k||_{n\times 1} = \min_i ||\mathbf{d}_{k,i}||. \tag{5}$$

In each round, each cluster must select one UE from the column vector of minimum distances $\mathbf{M}$. First, the UE with the minimum distance is selected by the cluster, and this cluster does not select another UE in this round.

$$\mathbf{u}_i \in C_k, \text{ if } \min ||\mathbf{m}_k|| = d_{k,i}. \tag{6}$$

The value of the kth row in the column vector $\mathbf{M}$ is set to ∞ to prevent this value from being selected again as the minimum. The distance values of UE i in the distance matrix $\mathbf{D}$ are set to ∞ to prevent the UE from being assigned to another cluster.

The pseudocode for the modified k-means cluster assignment step is shown in Algorithm 1.

Algorithm 1 Modified k-means cluster assignment step.

```
repeat
  repeat
    Construct vector $\mathbf{M}$ from distance matrix $\mathbf{D}$ using Equation (5)
    Find minimum in $\mathbf{M}$ and assign UE $i$ to cluster $k$
  until All clusters have a UE
until All clusters have minimum number of UEs assigned
```

Next, when all clusters have collected the minimum number of UEs, the regular k-means cluster assignment is processed for the rest of the UEs still unassigned to a cluster:

$$\mathbf{u}_i \in C_k, \text{ if } \min_k(||\mathbf{d}_{k,i}||) = d_{k,i}. \tag{7}$$

After cluster assignment, the new centroids are calculated as the mean of all points in the cluster. The convergence criterion is that the new centroid values remain the same as at the end of the previous iteration, as in the original k-means algorithm.

The complete proposed UE clustering algorithm is shown in Algorithm 2.

Algorithm 2 Proposed algorithm for UE clustering.

```
Initialization: Set $K$ random centroids
repeat
  Calculate distance matrix $\mathbf{D}$
  repeat
    Cluster assignment with modified k-means (Algorithm 1)
  until All clusters have minimum number of UEs assigned
  if unassigned UEs exist then
    repeat
      Cluster assignment with original k-means (Equation (7))
    until All UEs have cluster assigned
  end if
  Calculate new centroids
until convergence
```

3.3. Proposed Group Bandwidth-Allocation Algorithm

The slice bandwidth is allocated to the UE groups proportionally to the number of UEs in a group and the group BLER and taking into account that no subslice has a size smaller than the minimum subslice size criteria. Thus, the slice bandwidth part is allocated in three steps:

$$N^{(RB)} = N_1^{(RB)} + N_2^{(RB)} + N_3^{(RB)} \tag{8}$$

and for subslices

$$N_k^{(RB)} = N_{1,k}^{(RB)} + N_{2,k}^{(RB)} + N_{3,k}^{(RB)}. \tag{9}$$

Initially, all RBs are divided for the subslices and each subslice receives at least the initial number of RBs per UE. The initial allocation factor is calculated as follows:

$$P_1 = \frac{(N^{(RB)}/K) \cdot (N^{(RB)}/S_{min})}{N^{(RB)}/N^{(UE)}}, \tag{10}$$

which can be simplified to

$$P_1 = \frac{1}{N^{(UE)}} \cdot \frac{K \cdot S_{min}}{N^{(RB)}} \tag{11}$$

and the initial number of RBs of a subslice is

$$N_{1,k}^{(RB)} = \lfloor N_k^{(UE)} \cdot P_1 \rfloor. \tag{12}$$

The second allocation is proportional to the group BLER and the number of UEs in a cluster. The cluster with the better (smaller) BLER obtains fewer RBs per UE, and the cluster with the worse (larger) BLER obtains more RBs per UE in a cluster.

The BLER factor for a group is calculated from the group BLERs calculated using Equation (2) and the maximum BLER of group, which obtains a BLER factor of 1:

$$f_{BLER_k} = \frac{BLER_{C_k}}{\max(BLER_{C_k})}. \tag{13}$$

The UE groups are not equal in size; therefore, the allocation factor P_2, which notes the RBs per UE proportional to BLER, is required. This is calculated from the group BLER factor and the number of UEs in a group.

$$P_2 = \frac{N^{(RB)} - N_1^{(RB)}}{\sum\limits_{k=1}^{K} f_{BLER_k} \cdot N_k^{(UE)}}. \tag{14}$$

All required factors are calculated, and the remaining RBs from the first allocation are allocated to the groups using the following equation:

$$N_{2,k}^{(RB)} = \lfloor N_k^{(UE)} \cdot f_{BLER_{C_k}} \cdot P_2 \rfloor. \tag{15}$$

Finally, still too-small subslices will receive additional RB:

$$N_{3,k}^{(RB)} = \begin{cases} 1 & \text{if } (N_{1,k}^{(RB)} + N_{2,k}^{(RB)}) < S_{min}, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

The pseudocode of the bandwidth-allocation algorithm for the UE groups is shown in Algorithm 3:

Algorithm 3 Proposed RB allocation for UE groups.

```
Calculate initial allocation factor $P_1$ using Equation (11)
Calculate RBs to subslices proportional to number of UEs in a cluster, $N_{1,k}^{(RB)}$ (Equation (12))
if $N^{(RB)} - N_1^{(RB)} > 0$ (unallocated RBs exist) then
    Second allocation proportional to group BLER:
    Calculate group BLERs (Equation (2))
    Calculate group BLER factors (Equation (13))
    Calculate allocation factor $P_2$ (Equation (14))
    Calculate RBs to subslices proportional to group BLER and number of UEs in a cluster, $N_{2,k}^{(RB)}$ (Equation (15))
end if
if $N^{(RB)} - N_1^{(RB)} - N_2^{(RB)} > 0$ (unallocated RBs exist) then
    Add RB to too-small subslices, Equation (16)
end if
```

3.4. Proposed Subslicing Algorithm

The full UE clustering with the bandwidth-allocation algorithm is as follows: first, the minimum cluster size for UE clustering is calculated using Equation (1). Then, UEs are clustered using Algorithm 2. Finally, the slice RBs are allocated to the UE groups using Algorithm 3.

4. Slice Performance Evaluation

We used the same simulation methodology as our previous work [2], described in Section 2.2, where the MATLAB tool is utilized to simulate individual subslices with specified parameters, including the number of RBs and UEs. In the first step, the slice UEs are all in one subslice, and all slice bandwidth is allocated to this one subslice. The slice performance and the block error ratio (BLER) for each UE are measured.

When the minimum subslice sizes are 73 and 37 RBs, three and seven subslices can be created, respectively. Otherwise, when there are more subslices, one or more subslices must be too small to degrade the slice performance. Smaller subslice sizes are tested to verify their effects on the poor slice performance.

4.1. Simulation Setup

Simulations are used to compare the slice performance if it is subsliced using equal UE grouping, k-means UE clustering, and the proposed subslicing algorithm. The equal-grouping algorithm groups the UEs into groups of as equal a size as possible and allocates RBs to the UE groups proportionally to the number of UEs in a group. This does not consider the diverse bandwidth requirements of UEs with different BLERs. K-means clusters the UEs into clusters of different sizes, and group-specific bandwidth allocation is not implemented. A subslice that is too small has a poor performance, which affects the slice performance, and the SLA of the UEs in a subslice that is too small will not be satisfied. The subslicing using k-means could be similar to [5], where the features are selected using SVM and the UEs are clustered using k-means. Although the number of features (clusters) is determined using SVM, the same weakness of k-means still remains: the cluster may be too small for achieving a good slice performance. The proposed algorithm uses modified k-means to create clusters that are not smaller than the minimum size requirement and allocates the bandwidth to the UE group proportionally to the number of UEs in a group and the group BLER. The subslices will not be too small, and the group BLER is considered in bandwidth allocation to the group. The slice does not require additional bandwidth to improve its performance.

The test cases are presented in Table 5. The slice is described as a bandwidth resource in RBs. The slice is allocated 275 RBs, as is the largest bandwidth 50 MHz or 100 MHz if using a subcarrier spacing of 15 or 30 kHz, respectively. If a larger bandwidth is needed, then carrier aggregation should be used. The slice resources are used by the UEs, and one UE is assumed to consume one RB, as in the previous simulations. The test cases include good-BLER, medium-BLER, and poor-BLER UEs. In order to obtain different BLER values with the channel model used, the UEs are located at different distances from the gNB.

Table 5. Simulation settings.

Parameter	Value
Slice bandwidth	275 RBs (49.5 MHz)
Subcarrier spacing	15 kHz
Number of UEs is the slice	275
Number of subslices and minimum subslice sizes	3, if $S_{min} = 73$ 7, if $S_{min} = 37$ 14, if $S_{min} = 19$ 25, if $S_{min} = 11$ 68, if $S_{min} = 4$
Parameter to cluster UEs	BLER
UE distance from gNB (m): all good-BLER UEs, all medium-BLER UEs, all poor-BLER UEs	. 1–275 1001–1275 6001–6275
UE rate requirements	UL 500 kbps, DL 667 kbps
UE packet sizes	{1500 B, 40 B}

From the previous section, it is clear that the subslice size should be at least 37 RBs to achieve a sufficient performance. For the simulations, the following numbers of subslices and minimum subslice sizes are selected for the proposed algorithm: three ($S_{min} = 73$, the highest goodput expected), seven ($S_{min} = 37$, the lowest utilization expected), 14 ($S_{min} = 19$), 25 ($S_{min} = 11$), and 68 ($S_{min} = 4$). The last three subslice sizes are simulated to verify their effects on the poor slice performance. K-means is not used if 68 subslices need to be created. MATLAB cannot simulate $BWP < 4$ RBs. Dividing 275 RBs into 68 BWPs, the subslices are at least four RBs, and only three subslices can be five RBs. It is difficult to achieve many small clusters of almost equal size by clustering.

The subslicing framework works on top of the scheduler, and the direct signal-quality parameters may not be available. The parameter to cluster UEs was selected to use BLER because it characterizes the signal quality for the UE better than the UE distance from the BS. Similar UEs based on their signal quality in the same subslice could have fairness in transmission. Any other parameter that characterizes the need for additional bandwidth for packet retransmission can be used for UE clustering.

Other simulation parameters are listed in Table 3. Performance data is collected for each subslice and combined to obtain the slice performance data. Each simulation is performed ten times. The mean values and 95% confidence intervals are calculated using MATLAB and its functions mean and fitdist, respectively.

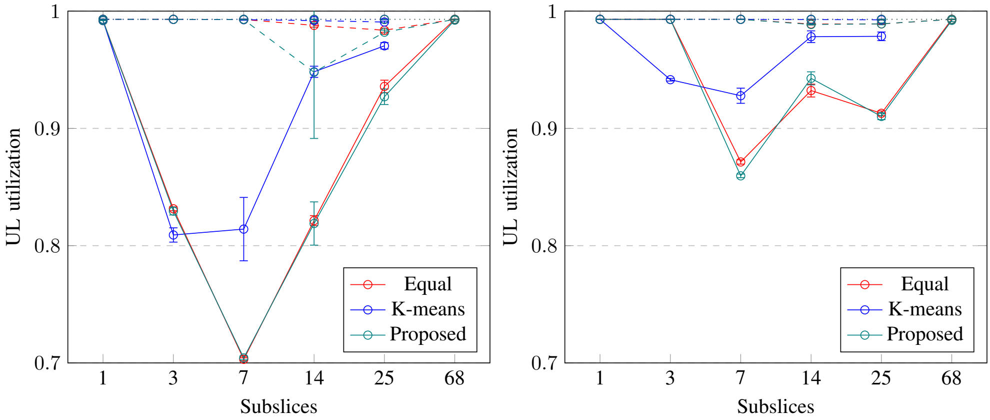
4.2. Results

The slice performance is measured with the slice bandwidth utilization and achieved slice throughput and goodput. In addition, the average slice BLER is collected. Finally, the slice performance improvement achieved by subslicing compared to not subslicing is discussed.

The horizontal axis of the graphs show how many subslices the slice has been subsliced into. The vertical axis shows the performance metric.

4.2.1. Slice Bandwidth Utilization

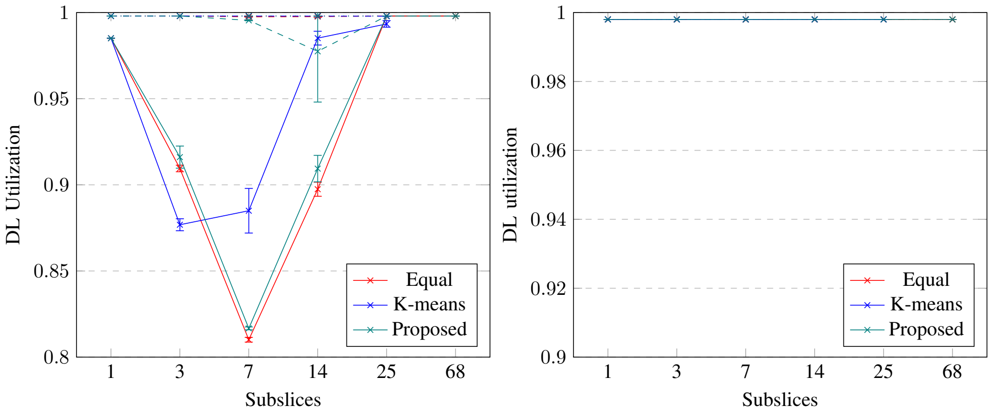
The slice bandwidth utilization in the UL is shown in Figure 6. If the slice is not subsliced, the UL bandwidth utilization is 100%. The utilization decreases when the slice is subsliced into a few subslices, but if there are more than seven subslices, then the utilization in UL increases back to 100%. The slice bandwidth utilization in the UL decreases to 70% if 1500-byte packets are used, whereas 40-byte packet utilization decreases to 85%. Equal UE grouping and the proposed subslicing algorithm decrease the slice utilization more than k-means UE clustering. If the UE BLER is worse (dashed and dotted lines), subslicing does not decrease the slice bandwidth utilization. To improve the slice bandwidth utilization in UL by up to 41% by subslicing, the slice should contain good-BLER UEs that use longer packets. Equal UE grouping and the proposed algorithm can be used to perform subslicing into no more than seven subslices with a minimum subslice size of 37 RBs.



(**a**) UL utilization (1500 B) (**b**) UL utilization (40 B)

Figure 6. Simulation results on utilization in UL. Error bars show a confidence interval of 95%. Solid lines show slices containing good-BLER UEs, dashed lines show slices containing medium-BLER UEs, dotted lines show slices containing poor-BLER UEs.

The slice bandwidth utilization in DL is shown in Figure 7. If the slice is not subsliced, the DL bandwidth utilization is close to 100%. The utilization decreases when the slice is subsliced into a few subslices, but if there are more than seven subslices, then the utilization in UL increases back to 100%. The slice bandwidth utilization in DL decreases to 80% if 1500-byte packets are used, whereas for 40-byte packets, the utilization does not decrease at all. Equal UE grouping and the proposed subslicing algorithm decrease the slice utilization more than k-means UE clustering. Similar to UL, if the UE BLER is not good, then subslicing does not decrease the slice bandwidth utilization in DL. To improve the slice bandwidth utilization in DL by up to 22% by subslicing, the slice should contain good-BLER UEs that use longer packets. Equal UE grouping and the proposed algorithm can be used to perform subslicing into no more than seven subslices with a minimum subslice size of 37 RBs.

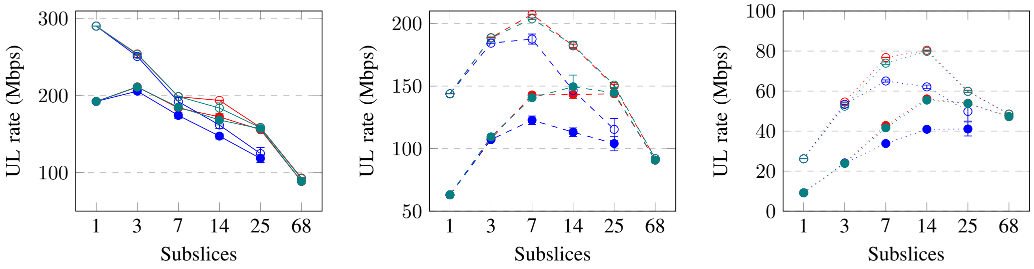


(**a**) DL utilization (1500 B) (**b**) DL utilization (40 B)

Figure 7. Simulation results on utilization in DL. Error bars show a confidence interval of 95%. Solid lines show slices containing good-BLER UEs, dashed lines show slices containing medium-BLER UEs, dotted lines show slices containing poor-BLER UEs.

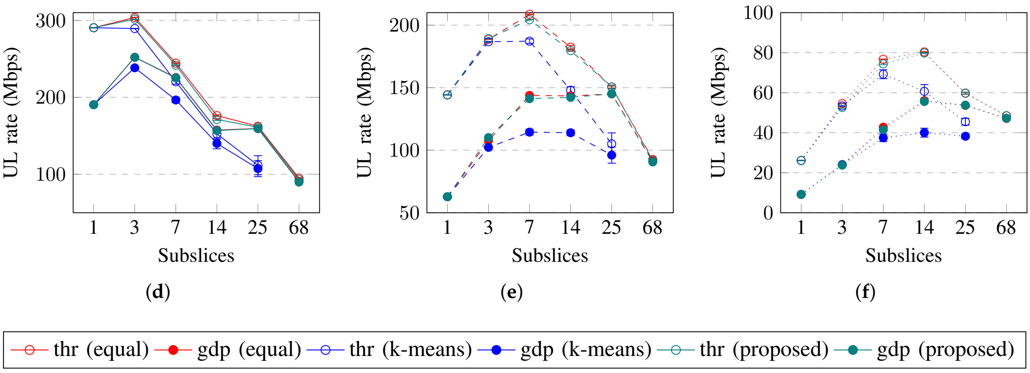
4.2.2. Slice Throughput and Goodput

The achieved throughput and goodput for a slice in the UL are shown in Figure 8. The slice throughput and goodput in the UL initially increase and then decrease when the slice divided into more subslices. The rates achieve maximum values when the slice is subsliced into three, seven, or 14 subslices if the slice contains good-BLER, medium-BLER, or poor-BLER UEs, respectively. Subslicing increases the goodput even more if the packet size is short, for example, 40 B. The achieved rates are higher when subslicing is performed using equal grouping or the proposed algorithm. Subslicing increased the achieved UL rates by up to 9%, 58%, and 84% if the UE BLER is good, medium, or poor, respectively. To increase the slice goodput in the UL by subslicing, the equal UE grouping or proposed algorithm should be used to subslice the slice into three, seven, or 14 subslices. Subslicing improves the slice goodput more if the UEs have worse BLER and use short packets.



(**a**) (**b**) (**c**)

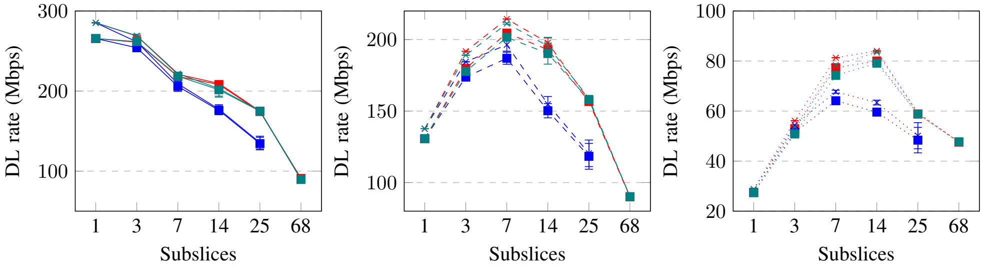
Figure 8. *Cont.*



(**d**) (**e**) (**f**)

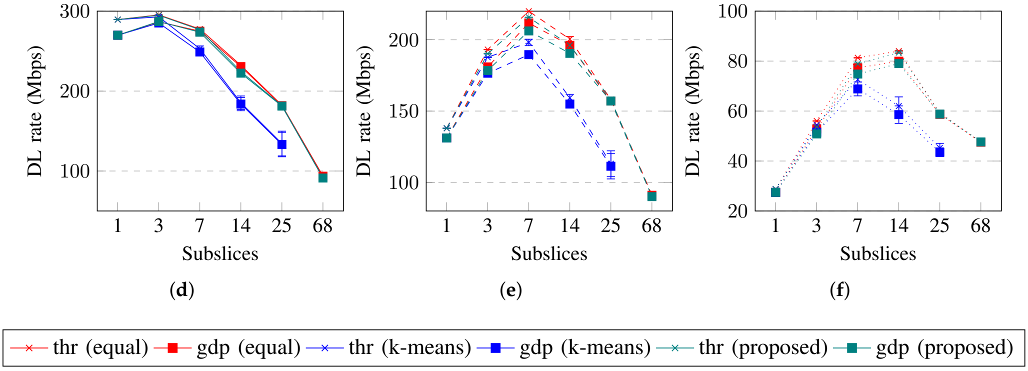
Figure 8. Simulation results on throughput (thr) and goodput (gdp) in uplink (UL). Error bars show a confidence interval of 95%. Solid lines show slices containing good-BLER UEs, dashed lines show slices containing medium-BLER UEs, dotted lines show slices containing poor-BLER UEs. (**a**) UL capacities (1500 B, good-BLER). (**b**) UL capacities (1500 B, medium-BLER). (**c**) UL capacities (1500 B, poor-BLER). (**d**) UL capacities (40 B, good-BLER). (**e**) UL capacities (40 B, medium-BLER). (**f**) UL capacities (40 B, poor-BLER).

The achieved throughput and goodput for a slice in DL are shown in Figure 9. The slice throughput and goodput in the DL change less than those in the UL. Similarly, in DL, the rates achieve maximum values when the slice is subsliced into three, seven, or 14 subslices if the slice contains good-BLER, medium-BLER, or poor-BLER UEs, respectively. Subslicing increases the DL goodput if the packet size is short and when subslicing is performed using equal grouping or the proposed algorithm. Subslicing increases the achieved DL rates by up to 6%, 38%, and 66% if the UE BLER is good, medium, and poor, respectively. To increase the slice goodput in DL by subslicing, equal UE grouping or the proposed algorithm should be used to subslice the slice into three, seven, or 14 subslices. Subslicing does not improve the slice goodput in DL if the UEs use long packets and have good BLER.



(**a**) (**b**) (**c**)

Figure 9. *Cont.*

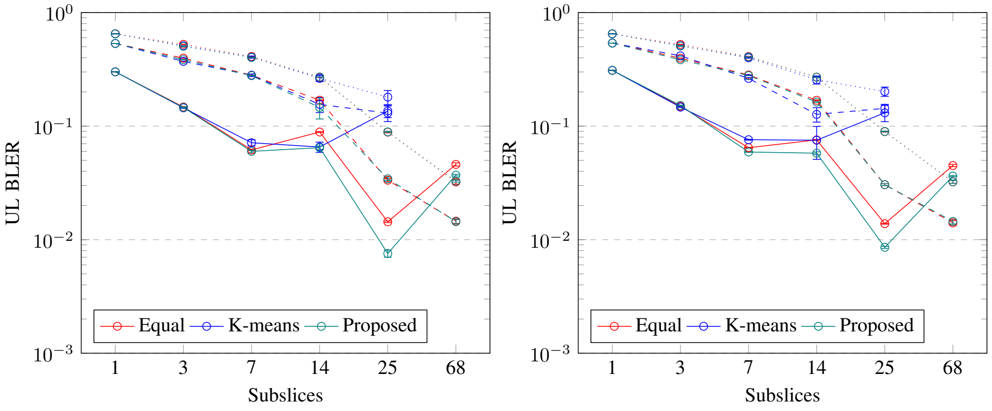


(**d**) (**e**) (**f**)

Figure 9. Simulation results on throughput (thr) and goodput (gdp) in downlink (DL). Error bars show a confidence interval of 95%. Solid lines show slices containing good-BLER UEs, dashed lines show slices containing medium-BLER UEs, dotted lines show slices containing poor-BLER UEs. (**a**) DL capacities (1500 B, good-BLER). (**b**) DL capacities (1500 B, medium-BLER). (**c**) DL capacities (1500 B, poor-BLER). (**d**) DL capacities (40 B, good-BLER). (**e**) DL capacities (40 B, medium-BLER). (**f**) DL capacities (40 B, poor-BLER).

4.2.3. Slice BLER

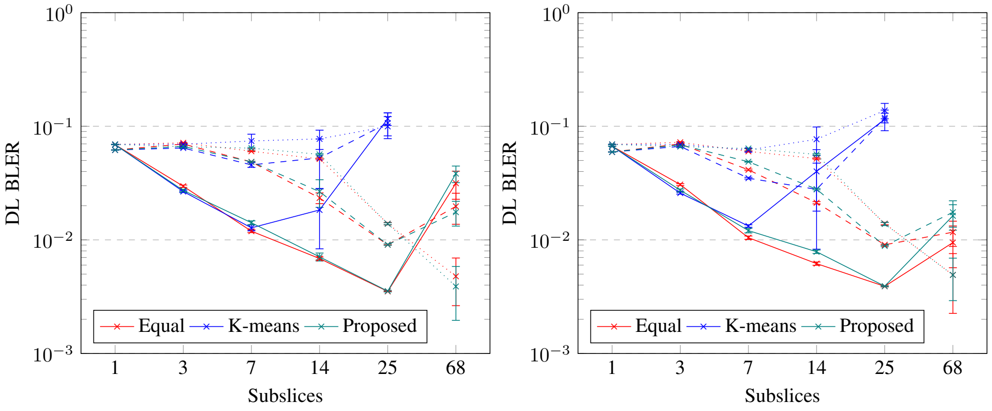
The slice BLER is the average of the UE BLERs. The slice BLER is shown in Figures 10 and 11 for the UL and DL, respectively. The slice BLER in both the UL and DL is decreasing if the slice is subsliced into up to seven subslices. The slice BLER is similar for both packet sizes, except for DL with k-means UE clustering, in which short packets and 14 subslices of DL BLER are the lowest. The slice BLER in UL is a minimum if the slice is subsliced into two, seven, or 25 using the proposed algorithm; however, if 14 subslices are used, k-means achieves the best UL BLER. The slice BLER in DL is a minimum if the slice is subsliced into 25 using the proposed algorithm or equal UE grouping; however, if k-means UE clustering is used, then seven or 14 subslices achieves the best DL BLER with 1500-B and 40-B packet sizes, respectively. Subslicing enables a decrease in the BLER, especially for slices with good-BLER UEs; however, to decrease the slice BLER, if the slice contains medium- or poor-BLER UEs, more subslices are necessary. The slice BLER in the UL decreases with subslicing. If seven or fewer subslices are needed for good-BLER UEs, then using the proposed algorithm, the BLER in the UL improves the most. The slice BLER in DL can be decreased by subslicing into at least seven subslices if the slice contains good-BLER UEs. If equal UE grouping and the proposed algorithm are used for subslicing a slice that contains good-BLER UEs, the number of subslices can be up to 25. Subslicing can improve the slice BLER if equal UE grouping or the proposed algorithm is used to create 25 or 68 subslices for slices that contain medium-BLER or poor-BLER UEs, respectively. The slice BLER in UL is above 0.1 and it decreases below this value if the slice contains all good-BLER UEs and is subsliced into seven subslices. If the slice contains UEs with worse BLER, then $BLER < 0.1$ is when 25 subslices are created using equal grouping or the proposed algorithm. In DL, the BLER is mostly below 0.1, except when 25 subslices are created using k-means UE clustering.



(**a**) UL BLER (1500 B)

(**b**) UL BLER (40 B)

Figure 10. Simulation results on UL BLER. Error bars show a confidence interval of 95%. Solid lines show slices containing good-BLER UEs, dashed lines show slices containing medium-BLER UEs, dotted lines show slices containing poor-BLER UEs.



(**a**) DL BLER (1500 B)

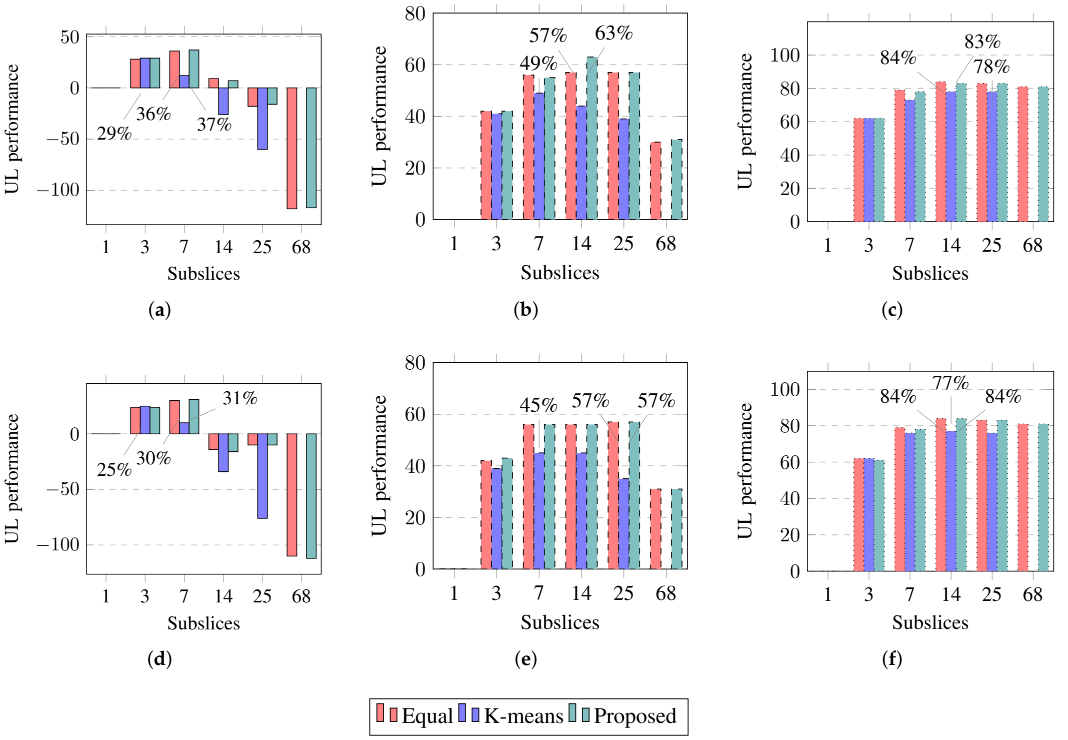
(**b**) DL BLER (40 B)

Figure 11. Simulation results on DL BLER. Error bars show a confidence interval of 95%. Solid lines show slices containing good-BLER UEs, dashed lines show slices containing medium-BLER UEs, dotted lines show slices containing poor-BLER UEs.

4.2.4. Slice Performance Improvement

The slice performance is improved by subslicing if the slice bandwidth utilization decreases and achieved slice goodput increases. The percentage decrease in utilization is added to the percentage increase in goodput and compared to a slice not subsliced; that is, the number of subslices is one.

The slice performance improvement in the UL is shown in Figure 12. The results show that the slice performance in UL can be improved another 6% more if a longer packet size is used. Equal UE grouping and the proposed algorithm improve the slice performance more than k-means UE clustering. Subslicing improves the slice performance in UL by up to 37%, 63%, or 84% if the slice contains good-BLER, medium-BLER, or poor-BLER UEs, respectively. To improve the slice performance in the UL by subslicing, equal UE grouping or the proposed algorithm should be used to subslice the slice into seven or 14 subslices. If UEs have worse BLER, more subslices can be recommended.

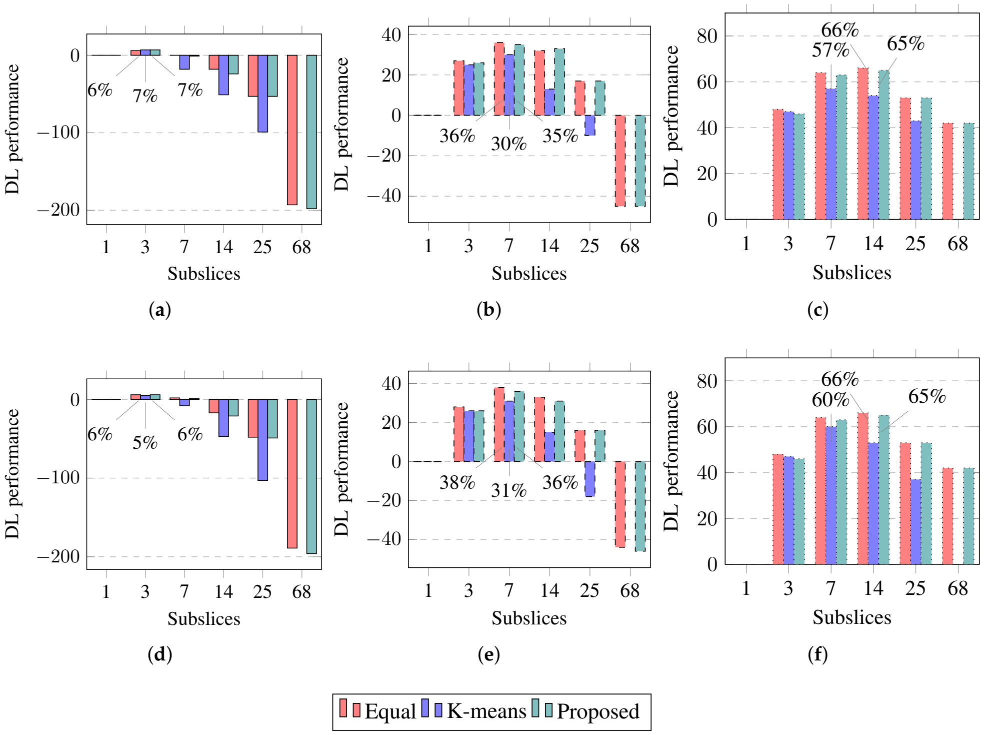


(**a**) (**b**) (**c**)

(**d**) (**e**) (**f**)

Figure 12. Simulation results on performance improvement percentages in UL compared to slice not subsliced. Solid lines show slices containing good-BLER UEs, dashed lines show slices containing medium-BLER UEs, dotted lines show slices containing poor-BLER UEs. (**a**) UL performance (1500 B, good-BLER). (**b**) UL performance (1500 B, medium-BLER). (**c**) UL performance (1500 B, poor-BLER). (**d**) UL performance (40 B, good-BLER). (**e**) UL performance (40 B, medium-BLER). (**f**) UL performance (40 B, poor-BLER).

The improvement in the slice performance in DL is shown in Figure 13. The slice performance in DL can be improved to a lesser extent than that in UL. The slice performance improvement in DL is similar for both packet sizes and is similar to UL and equal UE grouping, and the proposed algorithm improves the slice performance more than k-means UE clustering. Subslicing improves the slice performance in DL by up to 7%, 38%, or 66% if the slice contains good-BLER, medium-BLER, or poor-BLER UEs, respectively. To improve the slice performance in DL by subslicing, equal UE grouping or the proposed algorithm should be used to subslice the slice into three, seven, or 14 subslices if the slice contains good-BLER, medium-BLER, or poor-BLER UEs, respectively.



(**a**) (**b**) (**c**) (**d**) (**e**) (**f**)

Figure 13. Simulation results on performance improvement percentages in DL compared to slice not subsliced. Solid lines show slices containing good-BLER UEs, dashed lines show slices containing medium-BLER UEs, dotted lines show slices containing poor-BLER UEs. (**a**) DL performance (1500 B, good-BLER). (**b**) DL performance (1500 B, medium-BLER). (**c**) DL performance (1500 B, poor-BLER). (**d**) DL performance (40 B, good-BLER). (**e**) DL performance (40 B, medium-BLER). (**f**) DL performance (40 B, poor-BLER).

4.2.5. Algorithm Performance

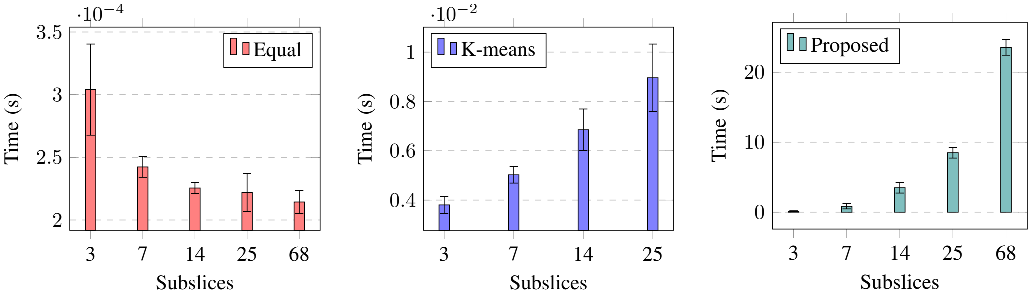
The performance of the proposed subslicing algorithm was compared with that of equal UE grouping and k-means UE clustering. The evaluation of the subslicing algorithms was based on the measurement of the slice performance improvement achieved and the computational time required for the algorithm to calculate the subslice settings.

The performance improvement percentages for UL and DL, averaged across all UE types, are presented in Table 6. The results indicate that creating seven subslices using equal grouping or the proposed algorithm achieves the greatest improvement in slice performance. On average, subslicing improved the slice performance by over 40% compared to a non-subsliced slice.

To evaluate the complexity of different subslicing algorithms, we measured the time it took to run the MATLAB implementation. Each algorithm ran 60 times, with ten runs per test case. Figure 14 shows the measured average time, and the error bars indicate a 95% confidence interval. When using equal grouping, the time required to calculate the subslice configuration did not depend on the number of subslices. However, other algorithms took more time as the number of subslices increased. Although the proposed algorithm required the most time, the subslices it created outperformed those created using k-means UE clustering.

Table 6. The average slice performance improvement summed for UL and DL and compared with that when the slice was not subsliced.

Algorithm	3 Subslices	7 Subslices	14 Subslices	25 Subslices	68 subslices
Equal	35.3	45	36.5	24.2	−32.7
K-means	34.6	34.8	18.4	−4.9	N/A
Proposed	34.8	44.3	35.5	24.3	−33.8



(**a**) Equal grouping (**b**) K-means clustering (**c**) Proposed subslicing algorithm

Figure 14. Measured time it takes to create sublices—group/cluster slice UEs and subpartition slice bandwidth. Error bars show a confidence interval of 95% from 60 runs.

In conclusion, subslicing can be recommended to improve the slice performance. Subslicing is more effective in reducing the slice bandwidth utilization with long packets and increasing the slice goodput with short packets. The UL benefits more from subslicing due to a greater reduction in the UL bandwidth utilization. Both random UE grouping and the proposed algorithm are suitable for subslicing, but k-means UE clustering creates a set of clusters where some clusters are too small, which results in a poor subslice performance and degrades the slice performance. Subslicing is more effective in improving the slice performance in the UL and for UEs whose BLER is not good. The recommended number of subslices to be created is higher if the slice contains UEs whose BLER is worse.

Dividing a network slice into suitably sized subslices can have positive system implications, as it can improve the slice performance and provide service categories with specific resource allocations to satisfy particular requirements. However, subslicing also has negative implications. For example, it can increase the network management complexity and require additional resources to calculate the subslice configurations.

5. Conclusions

In this paper, subslicing was investigated as a method to improve the RAN slice performance. The present literature does not evaluate the number and size of subslices required to achieve a slice performance improvement in 5G-NR. The slice bandwidth was subpartitioned and allocated to smaller groups of slice UEs. The subslices were simulated individually, and the performance data were combined to represent the slice performance.

Our work demonstrates the positive effect of subslicing on slice performance. Moreover, our work has determined the criteria to achieve a slice performance improvement. The benefit of subslicing is the efficient use of radio resources; however, it requires computing and storage resources to create subslices.

The simulations were performed with all UEs having similar rate requirements and BLER. This enables the evaluation of how the performance of slice UEs can be improved by subslicing. However, the subslicing algorithm can be improved for a realistic case in which the UEs have different BLERs and requested rates. Then, the subslices for the UEs with worse BLERs can be smaller.

The results show that the slice performance depends on the number of subslices and the subslice performance depends on its size. The minimum subslice size requirement ranged from 11 to 73 RBs. The lower the UE BLER, the higher the minimum subslice size. The number of subslices with which the slice performance can be improved is higher if the BLER of the UEs is higher.

Future work will add the decision of subslicing to the toolbox for slice modification to improve the slice performance without additional bandwidth.

Author Contributions: Conceptualization, M.K.; methodology, M.K.; software, M.K.; validation, M.K., I.M., and M.M.A.; formal analysis, M.K.; investigation, M.K.; resources, M.K.; data curation, M.K.; writing—original draft preparation, M.K.; writing—review and editing, M.K., I.M., and M.M.A.; visualization, M.K.; supervision, I.M. and M.M.A.; project administration, M.M.A.; funding acquisition, M.M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work has received funding partly from the European Union's Horizon 2020 Research and Innovation Program under Grant 951867 '5G-ROUTES' and Grant 101058505 '5G-TIMBER'. This work in the project "ICT programme" was supported by the European Union through European Social Fund, and TAR16013 Center of Excellence 'EXCITE IT'. This material reflects only the authors view and the EC Research Executive Agency is not responsible for any use that may be made of the information it contains.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. *TS 28.530*; Management and Orchestration; Concepts, Use Cases and Requirements; Technical Specification; 3GPP: Sophia Antipolis Cedex, France, 2020. Available online: https://www.3gpp.org/DynaReport/28530.htm (accessed on 7 September 2022).
2. Kulmar, M.; Muursepp, I.; Alam, M.M. The Impact of RAN Slice Bandwidth Subpartitioning on Slice Performance. In Proceedings of the 2022 International Wireless Communications and Mobile Computing (IWCMC), Dubrovnik, Croatia, 30 May–3 June 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 419–424. . [CrossRef]
3. *TS 23.501*; System Architecture for the 5G System (5GS); Stage 2; Technical Specification; 3GPP: Sophia Antipolis Cedex, France, 2023. Available online: https://www.3gpp.org/DynaReport/23501.htm (accessed on 10 January 2023).
4. Mohammedali, N.A.; Kanakis, T.; Al-Sherbaz, A.; Agyeman, M.O. Performance Evaluation for End-to-End Slice Management in 5G/B5G Cellular Networks. In Proceedings of the 2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 22–24 September 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6. [CrossRef]
5. Singh, S.K.; Salim, M.M.; Cha, J.; Pan, Y.; Park, J.H. Machine Learning-Based Network Sub-Slicing Framework in a Sustainable 5G Environment. *Sustainability* **2020**, *12*, 6250. [CrossRef]
6. Hasegawa, G.; Hasegawa, S.; Arakawa, S.; Murata, M. UONA: User-Oriented Network slicing Architecture for beyond-5G networks. In Proceedings of the ICC 2021—IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6. [CrossRef]
7. Ravindran, S.; Chaudhuri, S.; Bapat, J.; Das, D. Isolation-based Sub-Slices for Throughput Optimization in 5G Radio Access Network. In Proceedings of the 2019 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 26–27 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6. [CrossRef]
8. Ravindran, S.; Chaudhuri, S.; Bapat, J.; Das, D. EESO: Energy Efficient System-resource Optimization of Multi-Sub-Slice-Connected User in 5G RAN. In Proceedings of the 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2–4 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6. [CrossRef]
9. Ravindran, S.; Chaudhuri, S.; Bapat, J.; Das, D. Efficient Service Allocation Scheduling Algorithms for 5G User Equipments in Slice-in-Slice Networks. In Proceedings of the 2021 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Hyderabad, India, 13–16 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 36–41. [CrossRef]
10. Kuklinski, S.; Tomaszewski, L. Key Performance Indicators for 5G network slicing. In Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 24–28 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 464–471. [CrossRef]
11. Salhab, N.; Falou, S.E.; Rahim, R.; Ayoubi, S.E.E.; Langar, R. Optimization of the implementation of network slicing in 5G RAN. In Proceedings of the 2018 IEEE Middle East and North Africa Communications Conference (MENACOMM), Jounieh, Lebanon, 18–20 April 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6. [CrossRef]
12. Han, Y.; Tao, X.; Zhang, X.; Jia, S. Hierarchical Resource Allocation in Multi-Service Wireless Networks with Wireless Network Virtualization. *IEEE Trans. Veh. Technol.* **2020**, *69*, 11811–11827. [CrossRef]

13. Adachi, F.; Takahashi, R.; Matsuo, H. Enhanced Interference Coordination and Radio Resource Management for 5G Advanced Ultra-dense RAN. In Proceedings of the 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), Antwerp, Belgium, 25–28 May 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–5. [CrossRef]
14. Dghais, W.; Souilem, M.; Chi, H.R.; Radwan, A.; Taha, A.E.M. Dynamic Clustering for Power Effective Small Cell Deployment in HetNet 5G Networks. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–5. [CrossRef]
15. Wang, H.; Zhao, H.; Wu, W.; Xiong, J.; Ma, D.; Wei, J. Deployment Algorithms of Flying Base Stations: 5G and Beyond With UAVs. *IEEE Internet Things J.* **2019**, *6*, 10009–10027. [CrossRef]
16. Thantharate, A.; Paropkari, R.; Walunj, V.; Beard, C. DeepSlice: A Deep Learning Approach towards an Efficient and Reliable Network Slicing in 5G Networks. In Proceedings of the 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 10–12 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 0762–0767. [CrossRef]
17. Addad, R.A.; Taleb, T.; Flinck, H.; Bagaa, M.; Dutra, D. Network Slice Mobility in Next Generation Mobile Systems: Challenges and Potential Solutions. *IEEE Netw.* **2020**, *34*, 84–93. [CrossRef]
18. MacQueen, J. Some Methods for Classification and Analysis of MultiVariate Observations. In Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability, Berkeley , CA, USA, 21 June–18 July 1965; pp. 281–297.
19. Sinaga, K.P.; Yang, M.S. Unsupervised K-Means Clustering Algorithm. *IEEE Access* **2020**, *8*, 80716–80727. [CrossRef]
20. Bradley, P.S.; Bennett, K.P.; Demiriz, A. *Constrained k-Means Clustering*; Technical Report; Microsoft publication MSR-TR-2000-65; 2000; p. 8. Available online: https://www.microsoft.com/en-us/research/publication/constrained-k-means-clustering/ (accessed on 27 May 2022).
21. Malinen, M.I.; Fränti, P. Balanced k-means for clustering. In *Structural, Syntactic, and Statistical Pattern Recognition*; S+SSPR 2014. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8621, pp. 32–41. [CrossRef]
22. *TS 28.535*; Management and Orchestration; Management Services for Communication Service Assurance; Requirements; Technical Specification; 3GPP: Sophia Antipolis Cedex, France, 2020. Available online: https://www.3gpp.org/DynaReport/28535.htm (accessed on 10 January 2023).
23. Chang, C.Y.; Nikaein, N. RAN runtime slicing system for flexible and dynamic service execution environment. *IEEE Access* **2018**, *6*, 34018–34042. [CrossRef]
24. MathWorks. NR Cell Performance Evaluation with Physical Layer Integration. 2022. Available online: https://se.mathworks.com/help/5g/ug/nr-cell-performance-evaluation-with-physical-layer-integration.html (accessed on 19 January 2022).
25. GSMA. *NG.116 Generic Network Slice Template v5.0*; Technical Report; GSMA: London, UK, 2021. Available online: https://www.gsma.com/newsroom/wp-content/uploads//NG.116-v5.0.pdf (accessed on 6 September 2021).
26. *TS 38.214 NR*; Physical Layer Procedures for Data; Technical Specification; 3GPP: Sophia Antipolis Cedex, France, 2021. Available online: https://www.3gpp.org/DynaReport/38214.htm (accessed on 16 August 2022).
27. Fjodorov, A.; Masood, A.; Alam, M.M.; Parand, S. 5G Testbed Implementation and Measurement Campaign for Ground and Aerial Coverage. In Proceedings of the 2022 18th Biennial Baltic Electronics Conference (BEC), Tallinn, Estonia, 4–6 October 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Appendix 3

III

M. Kulmar, I. Müürsepp, and M. M. Alam, “Enhanced Decision Mechanism for RAN Subslicing in Management Closed Control Loop”, pp. 175–181, Sep. 2023, DOI: `10.1109/FMEC59375.2023.10306223`

Enhanced Decision Mechanism for RAN Subslicing in Management Closed Control Loop

Marika Kulmar
Thomas Johann Seebeck Department of Electronics
Tallinn University of Technology
Tallinn, Estonia
marika.kulmar@taltech.ee

Ivo Müürsepp
Thomas Johann Seebeck Department of Electronics
Tallinn University of Technology
Tallinn, Estonia
ivo.muursepp@taltech.ee

Muhammad Mahtab Alam
Thomas Johann Seebeck Department of Electronics
Tallinn University of Technology
Tallinn, Estonia
muhammad.alam@taltech.ee

***Abstract*—5G radio access network (RAN) slicing enables better satisfaction of different quality of service (QoS) requirements than without network slicing. A handful of slice types are specified for various service verticals; however, the number and size of those slices is unspecified. Subslicing refers to grouping slice users into smaller groups, subpartitioning slice bandwidth, and allocating smaller bandwidth parts to smaller user groups. State of the art subslicing has been done to better satisfy the QoS requirements inside the service vertical. Slice performance improvement was not the purpose of subslicing, but the positive effect was noticeable. In this paper, the subslicing decision is done with the aim of improving slice performance. The decision mechanism for management closed control loop is proposed. The input dataset consists of 6 key performance indicators, namely slice bandwidth utilization, slice goodput (application level throughput) per one allocated resource block (RB), and slice block error ratio (BLER), both in uplink and downlink. This dataset is clustered, and the result is learned by a classifier to decide whether the slice is too large and should be split or too small and should be merged with another too small slice or subslice. The results show that by knowing the slice utilization and goodput per one allocated RB, the slice reconfiguration action regarding subslicing can be determined using machine learning tools.**

***Index Terms*—subslicing, performance, clustering, neural network**

I. INTRODUCTION

Network slicing in a 5G radio access network (RAN) guarantees service-level agreement (SLA) using logical networks on the physical infrastructure. These logical networks, called slices, can be configured to satisfy specific quality of service requirements and connectivity. There are no limitations on the number or size of slices set by standardization.

The slice provisioning solutions allocate sufficient resources to user equipments (UEs) admitted to the slice. In RAN, the radio spectrum resource is available in bandwidth parts (BWP) of a fixed size [1], but can be allocated to the slice with the granularity of one resource block (RB). The RB is defined in timescale as a slot time and in frequency scale as 12 subcarriers. The length of a slot time and width of a subcarrier depend on subcarrier spacing configuration [2].

Given complete slice performance data, that is, the slice performance at any slice size, the dependence of slice performance on slice size can be determined. Subsequently, slices that are too large can be split into subslices, and slices or subslices that are too small can be merged. This decision mechanism can be used in "Decide" step of management closed control loop specified in 3GPP TS 28.535 [3].

How can we evaluate which subslice is too small and which is too large? How can we determine which subslice can be split and which one can be merged with another subslice that is too small? In this paper, it is proposed the clustering of subslice performance data into three clusters to determine whether the subslice should be merged, split, or not changed. Subsequently, the result of the performance data clustering is learned as a classification to decide suitable actions to reconfigure slices and subslices to improve the performance of a slice and network. Subslicing enables to serve more UEs if no additional bandwidth is available or if carrier aggregation is not possible.

The remainder of this paper is organized as follows. In Section II, the related work on slice and subslice size described. The proposed slice performance data clustering algorithm is described in Section III. Section IV contains the learning of clustering result as a classification. Finally, Section VI concludes the paper.

II. RELATED WORK

In RAN slicing, optimized resource allocation determines the slice size. Optimized resource allocation to a slice is expected to result in the best slice performance. In [4], resources were allocated to subslices based on optimization, and the channel bandwidth remained constant. Their results showed that their proposed algorithm for resource allocation to subslices outperformed the others multiple times if the number of subslices was higher. However, the UE SLAs varied; thus, the effect of the number of subslices on the cell performance was not comparable. It is not known if the number of UEs

Author's version of the paper accepted in 2023 Eighth International Conference on Fog and Mobile Edge Computing (FMEC). DOI: 10.1109/FMEC59375.2023.10306223 ©2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works

and average SLA for a slice were set equal among all number of subslices the slice was divided into.

The subslicing presented in [5] includes UE features selected using a support vector machine (SVM) and UEs clustered by selected features using k-means. The number of subslices was determined by evaluating the clustering quality, and performance was not considered when creating subslices. UEs with more similar requirements should work in one subslice; however, if the number of UEs is low, then the subslice is small and can exhibit poor performance.

Second, it is questionable to measure the effect of subslicing on performance change if the planned use of capacity does not match the allocated resources properly, that is, using requested data rates that are not capable of consuming the given BWP, or if the sum rate of UEs is too variable in time to evaluate the slice performance at different slice sizes.

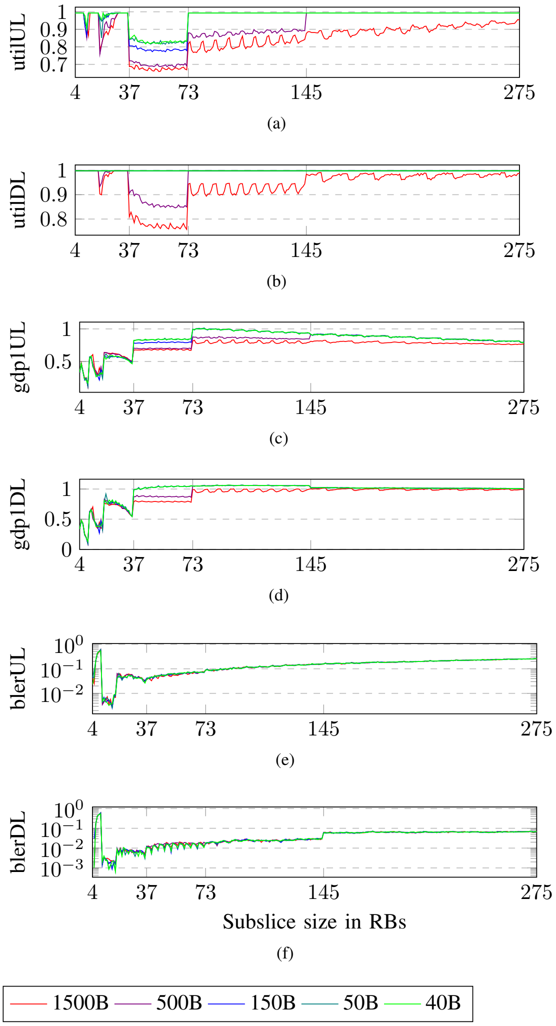
If there are many UEs in the slice, the slice BWP is large, and if there are few UEs in the slice, the slice BWP is small. However, BWPs that are too small cannot provide the same SLA satisfaction for the UE as if the UE is admitted to the slice that uses a large BWP. The performance of a small subslice can be reduced, as discussed in [6]; if the BWP is smaller than 20 physical RBs, then the different control blocks can puncture each other and increase the coupling loss in 5G NR.

III. PERFORMANCE DATA CLUSTERING

The aim of clustering the subslice performance data is to determine the action for subslice modification. The possible subslice modification options are “merge”, “no change” and “split”, thus three clusters are required. The execution of subslice modification is beyond the scope of this paper.

A. *Subslice performance data*

The subslice performance data were collected from the simulation results in the MATLAB 5G Toolbox system-level simulation tool called NR Cell Performance Evaluation with Physical Layer Integration [7]. The range of subslice size is 4-275 RBs (720 kHz—49.5 MHz, using subcarrier spacing of 15 kHz). Each RB in a subslice is expected to be consumed by one UE, which requests rates of 500 kbps in the UL and 667 kbps in the DL. The requested rate is served using packet sizes of 1500 bytes, 500, 150, 50, and 40 bytes. Each UE is located up to a distance of 173 m from the gNB and is expected to have good signal coverage. The key performance indicators (KPI) are subslice bandwidth utilization in UL (utilUL) and DL (utilDL), subslice goodput (application-level throughput) per one RB in UL (gdp1UL) and DL (gdp1DL), and the subslice average block error ratio (BLER) in UL (blerUL) and DL (blerDL). Further information regarding the dataset is provided in [8], section 2. Subslice performance improvement means that subslice utilization decreases, while subslice goodput increases. The subslice performance data is shown in Fig. 1



(a)

(b)

(c)

(d)

(e)

(f)

Fig. 1. Subslice performance data for all subslice sizes in range of 4-275 RBs, if packets of different sizes used. 6 KPIs: (a) bandwidth utilization in UL and (b)in DL, (c) goodput per one RB (Mbps) in UL and (d) in DL, (e) BLER in UL and (f) in DL.

B. *Clusterability evaluation*

Each data point in the dataset has values for six KPIs, subslice size in RBs, and packet size. To evaluate the clusterability [9] of subslice performance dataset, pairwise Euclidean distances were calculated for all data points. The histogram of all distances should be multimodal to expect the dataset to contain clusters. The histograms of pair-wise distances using combinations of KPIs are shown in Fig. 2. Histograms that characterize six or four KPIs look visually promising to be three-modal; thus, these should be explored further. The subslice performance data could not be clustered into three clusters by seven variables, consisting of the subslice size and six KPIs. The subslice size variable is used for the recognition of a data point, similar to the packet size variable in the dataset.

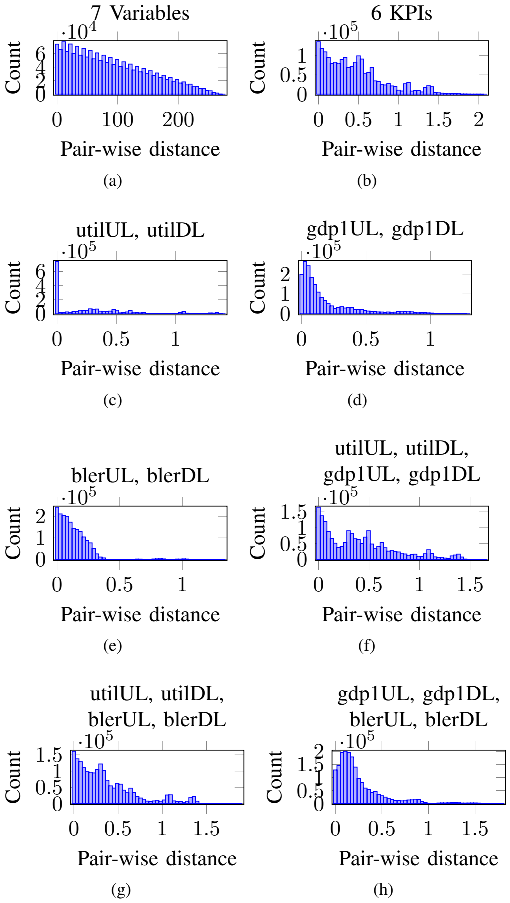


Fig. 2. The histogram of pair-wise distances of all data points in subslice performance dataset. Dimensions of each datapoint: (a) 6 KPIs and subslice size in RBs, (b) 6 KPIs, (c) bandwidth utilization in UL and DL, (d) goodput per one RB in UL and DL, (e) BLER in UL and DL, (f) utilization and goodput, (g) utilization and BLER, (h) goodput and BLER.

C. Evaluation of clustering results

Four types of clustering algorithms exist: density-based, distribution-based, centroid-based and hierarchical-based. One algorithm of each type was compared for suitability in clustering the subslice performance dataset into three clusters. K-means [10] is a centroid-based, simple, and popular clustering method. Gaussian Mixture Model (GMM) algorithm [11] is a distribution-based clustering method. Agglomerative hierarchy clustering algorithm [12] is a hierarchy-based clustering method. DBSCAN [13] is a density based method. The settings for DBSCAN are recommended in [14]: the minimum number of points, a parameter called $minPts$, and a small radius ϵ, that should contain the minimum number of points. Euclidean distance was used as the distance function. DBSCAN can identify outliers; however, a decision must be made for each data point.

To evaluate clustering results, three types of statistics were calculated. Davies-Bouldin index (DBI) [15] was calculated using

$$DBI = \frac{1}{k}\sum_{i=1}^{k} \max_{j\neq i}\{D_{i,j}\}, \tag{1}$$

where $D_{i,j}$ is the within-to-between cluster distance ratio for the ith and jth clusters. The smaller values indicate better clustering.

The Silhouette coefficient (SIL) [16] for each data point was calculated using

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)}, \tag{2}$$

where a_i average distance of point i to the points in the same cluster, b_i is the minimum average distance of point i to the points of all other clusters. High positive value means highly separable clusters.

The third statistic calculated was sum of entropies (SEN) of clusters by using

$$E = -\sum_{i=1}^{k} p_i \cdot \log(p_i), \tag{3}$$

where k is a number of clusters, and p_i is the proportion of the points in the region i. Large values of entropy indicate poor clustering behaviour [17].

TABLE I
EVALUATION OF CLUSTERS. *DBI WAS CALCULATED AFTER OUTLIERS EXCLUDED.

Variables	**DBI**	**SIL**	**SEN**
k-means			
6 KPIs	0.779	0.715	0.833
utilUL, utilDL, gdp1UL, gdp1DL	0.694	0.742	0.833
utilUL, utilDL, blerUL, blerDL	0.608	0.753	0.768
gdp1UL, gdp1DL, blerUL, blerDL	0.483	0.848	0.504
GMM			
6 KPIs	0.944	0.608	0.649
utilUL, utilDL, gdp1UL, gdp1DL	1.20	0.696	0.571
utilUL, utilDL, blerUL, blerDL	0.536	0.723	0.398
gdp1UL, gdp1DL, blerUL, blerDL	0.461	0.861	0.457
Agglomerative			
6 KPIs	0.409	0.566	0.0850
utilUL, utilDL, gdp1UL, gdp1DL	1.60	0.490	0.471
utilUL, utilDL, blerUL, blerDL	0.771	-0.348	0.0850
gdp1UL, gdp1DL, blerUL, blerDL	0.248	0.783	0.0850
DBSCAN			
6 KPIs	0.675*	0.569	0.427
utilUL, utilDL, gdp1UL, gdp1DL	0.602*	0.753	0.586
utilUL, utilDL, blerUL, blerDL	0.415*	0.775	0.0662
gdp1UL, gdp1DL, blerUL, blerDL	NaN*	NaN	0.0767

Better clustering is indicated if DBI is small, SIL is high positive value and SEN is small. The results are presented in Table I for all 6 KPIs and different combinations of 4 of those KPIs. The DBI was the smallest when Agglomerative clustering was used. Reducing the number of KPIs improves DBI. The SIL was the highest when k-means was used. Reducing the number of KPIs increased the SIL, except in the case of Agglomerative. The SEN was the lowest when Agglomerative clustering was used. If the performance data were clustered by utilization and goodput KPIs or utilization and BLER KPIs, then DBSCAN had the best statistical values. If the performance data were clustered by goodput and BLER KPIs, then GMM and Agglomerative had the best statistics, but DBSCAN could not be evaluated.

Considering the statistics, the best clustering is possible with Agglomerative hierarchical clustering, followed by k-means clustering, DBSCAN, and GMM. If the number of KPIs is reduced, then the statistics are improved for k-means and GMM, especially when BLER is included in the KPIs.

D. Visualization of clustering results

The aim of the visualization is to confirm that too small, too large, and suitably sized subslices are in three separate clusters, and that some boundaries of subslice sizes between the clusters are visible. The clustering results are presented in Fig. 3. The clusters are numbered automatically, but matching of the cluster to the decision is performed manually. The cluster which contains the smallest subslice sizes is matched to "merge" cluster. The cluster which contains the largest subslice sizes is matched to "split" cluster. A cluster which contains other subslice sizes is "no change" cluster.

Both algorithms, k-means and GMM, have similar results (see Fig. 3a and 3b); however, k-means has subslices that are too small to be smaller than 37 RBs, whereas GMM has subslices that are too small with sizes of 7-9 RBs. The removal of BLER KPI did not significantly change the clustering result (see Fig. 3c and 3d); however, if goodput was removed, too small and too large subslices were in the same cluster (Fig. 3e and 3f. Agglomerative (Fig. 3i) and DBSCAN (Fig. 3j) had most of the data points in one cluster.

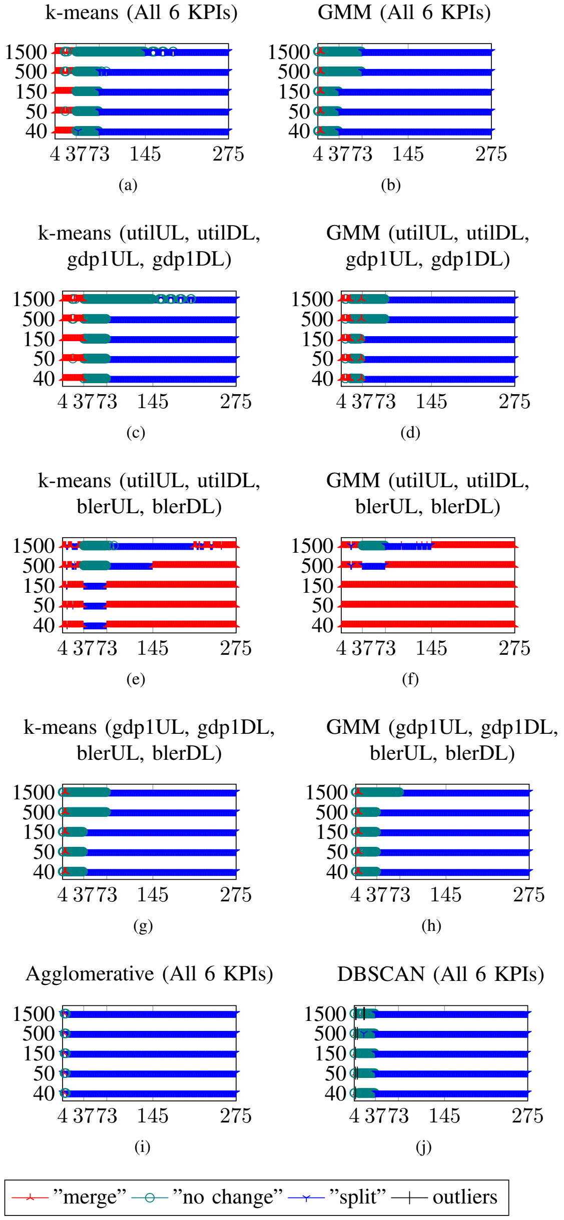
K-means had the second-best statistics, as shown in Table I, and it can identify subslices that are too small and suitable size better, as shown in the visualization of clusters in Fig. 3a and 3c.

IV. LEARNING THE CLUSTERING RESULTS FOR A CLASSIFICATION

The k-means clustering results were selected for learning by a classifier. The classification results of the two input datasets are compared: all six KPIs and four KPIs (utilUL, utilDL, gdp1UL, gdp1DL). This set of four KPIs had values of statistics not worse than the set of 6 KPIs, and had meaningful results in visualization.

A. Classifier

The machine learning tools suitable for multi-class classification are k nearest neighbour (KNN), supporting vector



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Fig. 3. Visualization of clustering results using different clustering algorithms and dimensions of a data point: (a) k-means (6 KPIs), (b) GMM (6 KPIs), (c) k-means (utilization and goodput), (d) GMM (utilization and goodput), (e) k-means (utilization and BLER), (f) GMM (utilization and BLER), (g) k-means (goodput and BLER), (h) GMM (goodput and BLER), (i)Agglomerative (6 KPIs), (j) DBSCAN (6 KPIs). The vertical axis shows used packet sizes in bytes, the horizontal axis shows subslice size in RBs.

machine (SVM) and neural network (NN). A neural network (NN) was selected as the tool for classification. NN has the flexibility and can work with large datasets.

The fully connected feedforward classifier NN consists of input layer, hidden layers, and output classification layer. The input layer contains 6 nodes, if 6 KPIs are used as inputs, and 4 nodes, if 4 KPIs are used as inputs. The output layer contains 3 neurons, one for each class (“merge”, “no change”, “split”).

The rectified linear unit (ReLU) activation function is suitable for the neurons in the hidden layers of the classifier NN. The number of hidden layers and number of neurons in the hidden layer need to be found by trials. The number of neurons in the hidden layer, N_h can be determined using some hints from [18] and options are shown in Table II. One and two hidden layers were used in the trials, and the number of neurons in each layer was set in the ranges of 1-22. The training, validation, and testing set ratios were 0.8, 0.1, and 0.1, respectively.

TABLE II
OPTIONS FOR NUMBER OF NEURONS IN HIDDEN LAYER, N_h, IF KNOWN NUMBER OF INPUTS N_i AND NUMBER OF OUTPUTS $N_o = 3$

	Number of neurons in hidden layer	
Condition [18]	**6 KPIs**	**4 KPIs**
$N_o \le N_h \le N_i$	3, 4, 5, 6	3, 4
$N_h = \frac{2}{3} \cdot N_i + No$	5	5
$N_h < 2 \cdot N_i$	12	8

To evaluate the quality of classification, cross entropy losses were calculated for training, validation, and testing. Cross entropy loss shows the probability distribution difference between the predicted and true classes for the multi-class classifier. The cross entropy loss is calculated using

$$L = \sum_{j=1}^{n} \bar{w}_j \frac{\log m_j}{K \cdot n}, \tag{4}$$

where $\bar{w}_j$ are normalized weights, m_j is a classification score predicted for a true class, K is the number of classes, n is the number of data points.

The minimum number of neurons and layers with small losses of training, validation, and testing were selected.

B. Results of NNs

Two input datasets were considered. The k-means clustering results if 6 KPIs were used, and if utilization and goodput without BLER were used.

Each number of neurons in hidden layers are tested 100 times. Mean and confidence interval of 95% are calculated. The results of NN are shown in Fig. 4. The NN performance is evaluated by calculation of a cross entropy loss for training, validation and testing, respectively.

For input dataset which contained 6 KPIs, if the number of neurons is 7 or more, then the losses are not decreasing further. Both the training and validation losses are lower if one hidden layer is used than with 2 hidden layers. This means

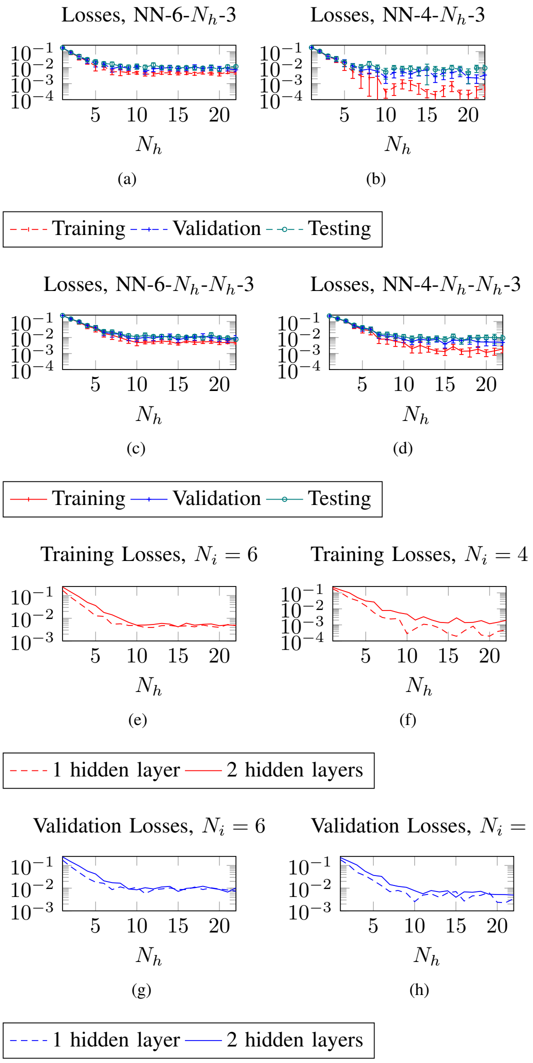


Fig. 4. Performance of a NN with one or two hidden layers (HL) and different number of neurons (N_h) in HL. (a) all losses for NN with one HL and input layer consisting of 6 nodes and (b) 4 nodes; (c) all losses for NN with two HLs and input layer consisting of 6 nodes and (d) 4 nodes, (e) training losses for NN with input layer consisting of 6 nodes and (f) 4 nodes; (g) validation losses for NN with input layer consisting of 6 nodes and (h) 4 nodes.

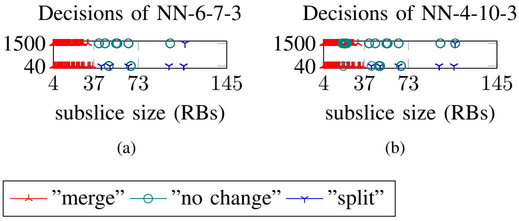
that one hidden layer with 7 neurons is sufficient configuration of classifier NN for subslice performance data with subslice sizes in the range of 4-275 RBs. The NN settings for a dataset of six KPIs are NN-6-7-3. The notation used, is the number of neurons in the input layer (6), the number of neurons in each hidden layer (1 hidden layer with 7 neurons), and the last number is the number of neurons in the output layer (3).

For input dataset which contained 4 KPIs, if the number of neurons is 10 or more, then the losses are not decreasing further. Similarly, all losses are lower when one hidden layer is used. The NN settings for a dataset of 4 KPIs are NN-4-10-3.

V. EXPERIMENTS

The constructed classifier, NN, was evaluated using a dataset of smaller subslices. Given the data of the six and four KPIs, the constructed NN with one hidden layer consisting of 7 and 10 neurons, respectively, determined the action for the subslice. The results are shown in Fig. 5. Generally, smaller subslices are classified as ”merge” and a few larger subslices are ”split”. If fewer KPIs are given as inputs, then some subslices smaller than 37 RBs are classified as ”no change”. When using 1500-byte packet size then the subslice size range of ”no change” is larger than if shorter packets are used.

It should be noted that these subslices were created by clustering UEs using k-means by UE BLER, and RBs were allocated proportionally to the number of UEs in the cluster. Second, the classifier NN does not know the subslice size in the RBs. Thus, the results are acceptable, and better classification was achieved if the input dataset contained six KPIs.



(a) (b)

Fig. 5. Classification experiments for smaller subslices using: (a) NN-6-7-3, (b) NN-4-10-3. The vertical axis shows used packet sizes in bytes, the horizontal axis shows subslice size in RBs.

VI. CONCLUSION

We have investigated the decision to modify slices using ML tools. The slice performance data is clustered into three clusters to match the slice or subslice configuration decisions. The clustering results were used as training data for the neural network, which determines the slice modification action based on the slice performance. The KPIs used as inputs were bandwidth utilization, goodput per allocated RB, and slice BLER, all for UL and DL. None of these KPIs can be omitted. This mechanism can help automatically decide subslices configuration. Further work is required to investigate the subslice performance dependence on the subslice size if the UEs are not in a good signal coverage.

ACKNOWLEDGMENT

This work has received funding partly from the European Union’s Horizon 2020 Research and Innovation Program under Grant 951867 ‘5G-ROUTES’ and Grant 101058505 ‘5G-TIMBER’. This work in the project “ICT programme” was supported by the European Union through European Social Fund, and TAR16013 Center of Excellence ‘EXCITE IT’.

REFERENCES

[1] GSM Association, “Generic Network Slice Template Version 8.0,” Tech. Rep., 2023, pp. 1–72, [Online]. Available: https://www.gsma.com/newsroom/resources/ng-116-generic-network-slice-template-v8-0/.

[2] 3GPP, “TS 38.211 - NR; Physical channels and modulation,” 2020, [Online]. Available: https://www.3gpp.org/DynaReport/38211.htm.

[3] 3GPP, “TS 28.535 - Management and orchestration; Management services for communication service assurance; Requirements,” Tech. Rep., 2020, [Online]. Available: https://www.3gpp.org/DynaReport/28535.htm.

[4] S. Ravindran, S. Chaudhuri, J. Bapat, and D. Das, “Isolation-based Sub-Slices for Throughput Optimization in 5G Radio Access Network,” in *2019 IEEE Int. Conf. Electron. Comput. Commun. Technol.*, IEEE, Jul. 2019, pp. 1–6, DOI: 10.1109/CONECCT47791.2019.9012928.

[5] S. K. Singh, M. M. Salim, J. Cha, Y. Pan, and J. H. Park, “Machine Learning-Based Network Sub-Slicing Framework in a Sustainable 5G Environment,” *Sustainability*, vol. 12, no. 15, p. 6250, Aug. 2020, DOI: 10.3390/su12156250.

[6] K. Hooli, P. Kinnunen, E. Tiirola, *et al.*, “Extending 5G to narrow spectrum allocations,” *IEEE J. Sel. Areas Commun.*, pp. 1–1, 2023, DOI: 10.1109/JSAC.2023.3273703.

[7] MathWorks, *NR Cell Performance Evaluation with Physical Layer Integration*, 2022, [Online]. Available: https://se.mathworks.com/help/5g/ug/nr-cell-performance-evaluation-with-physical-layer-integration.html (visited on 01/19/2022).

[8] M. Kulmar, I. Müürsepp, and M. M. Alam, “Heuristic Radio Access Network Subslicing with User Clustering and Bandwidth Subpartitioning,” *Sensors*, vol. 23, no. 10, p. 4613, May 2023, DOI: 10.3390/s23104613.

[9] A. Adolfsson, M. Ackerman, and N. C. Brownstein, “To cluster, or not to cluster: An analysis of clusterability methods,” *Pattern Recognit.*, vol. 88, pp. 13–26, Apr. 2019, DOI: 10.1016/j.patcog.2018.10.026.

[10] J. MacQueen, “Some Methods for Classification and Analysis of MultiVariate Observations,” in *Proc Berkeley Symp. Math. Stat. Probab.*, 1965, pp. 281–297.

[11] D. Reynolds, “Gaussian Mixture Models,” in *Encycl. Biometrics*, Boston, MA: Springer US, 2015, pp. 827–832, DOI: 10.1007/978-1-4899-7488-4_196.

[12] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley Seri. Wiley, 2009, ISBN: 9780470317488.

[13] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” *Proc. 2nd Int. Conf. Knowl. Discov. Data Min.*, 1996.

[14] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “DBSCAN Revisited, Revisited,” *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 1–21, Sep. 2017, DOI: 10.1145/3068335.

[15] D. L. Davies and D. W. Bouldin, “A Cluster Separation Measure,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979, DOI: 10.1109/TPAMI.1979.4766909.

[16] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987, DOI: 10.1016/0377-0427(87)90125-7.

[17] C. E. Shannon, “A Mathematical Theory of Communication,” *Bell Syst. Tech. J.*, vol. 27, no. 4, pp. 623–656, 1948, DOI: 10.1002/j.1538-7305.1948.tb00917.x.

[18] J. Heaton, *Introduction to Neural Networks with Java*. Heaton Research, 2008, ISBN: 9781604390087.

Appendix 4

IV

M. Kulmar, M. M. Alam, and I. Müürsepp, “Management Closed Control Loop Automation for Improved RAN Slice Performance by Subslicing”, *TechRxiv. December 18*, 2023, Submitted to IEEE Open J. Commun. Soc. (under review), DOI: `10.22541/techrxiv.170290948.88447519/v1`

IEEE ComSoc IEEE Open Journal of the Communications Society

Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version XX Month, XXXX.

Digital Object Identifier 10.1109/OJCOMS.2022.1234567

Management Closed Control Loop Automation for Improved RAN Slice Performance by Subslicing

MARIKA KULMAR*, **MUHAMMAD MAHTAB ALAM*** **SENIOR MEMBER, IEEE, AND IVO MÜÜRSEPP***

[1]Thomas Johann Seebeck Department of Electronics, Tallinn University of Technology, Tallinn, Estonia.

CORRESPONDING AUTHOR: Marika Kulmar (e-mail: marika.kulmar@ taltehc.ee).

This work has received funding partly from the European Union's Horizon 2020 Research and Innovation Program under Grant 951867 '5G-ROUTES' and Grant 101058505 '5G-TIMBER'. This work in the project "ICT programme" was supported by the European Union through European Social Fund, and TAR16013 Center of Excellence 'EXCITE IT'.

ABSTRACT Network slicing offers the potential to enhance service satisfaction and optimize resource utilization, particularly in scenarios where radio resources are limited. Subslicing has been shown to improve the slice performance. In this paper, the monitor-analyze-plan-execute-knowledge (MAPE-K)-type management closed control loop (MCCL) is implemented for slice performance improvement by subslicing. The subslicing can improve slice performance if the slice performance depends on the size of slice bandwidth part (BWP). For Plan function, the classifier neural network was trained to decide whether the subslice should be split, merged or not changed by their performance. The training data contains slice performance data of all possible subslice sizes. For Execute function, the subslice splitting algorithm was proposed, which clusters UEs by their block error ratio (BLER) and allocates bandwidth proportionally to group requested sum rate and group BLER. A realistic 5G new radio (NR) band serving a set of user equipments (UE) of different values of their BLER and requested rates was a setup of radio access network (RAN) slice simulated using MATLAB R2021b. Subslicing has reduced bandwidth utilization, and slice BLER while increased slice goodput (application-level throughput). Proposed subslice splitting algorithm when UEs are clustered by their achieved BLER, then the slice BLER reduces additional 20% and slice goodput increases up to additional 9% compared to no subslicing when UEs are clustered by their requested rates. This effect was larger for the uplink. In runtime scenarios for poor-BLER UEs the smaller subslices improve slice utilization and BLER, while larger subslices improve goodput.

INDEX TERMS 5G New Radio simulation, performance management, radio access network subslicing, reconfiguration automation

I. INTRODUCTION

Closed-loop network management plays a crucial role in meeting the growing demands for mobile communication, ensuring that cellular networks operate efficiently and effectively to provide a seamless experience for users. This automated management approach is often referred to as self-organizing networks (SON), zero-touch network and service management (ZSM), and management and orchestration (MANO). SON, in particular, relies on closed control loops (CCL) within the domains of self-configuration, self-optimization, and self-healing [1]. This innovation was first introduced for the management of cellular networks starting from 3GPP release 8 [2]. While SON primarily concentrates on automating cellular network-specific functions like configuration, optimization, and healing, ZSM extends these principles to a broader range of network and service management activities. MANO is a framework and set of functions used to manage and orchestrate the various components and services in a virtualized network environment. 3GPP has specified management closed control loop in TS 28.535 [3]. It is an automated process that continuously monitors, analyzes, and adjusts various aspects of a network to maintain or

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

improve its performance, quality, and efficiency. It operates in real-time or near-real-time, allowing for rapid responses to changing network conditions.

3GPP has specified a life cycle of network slice in TS 28.530 [4]. During the slice modification phase, managing slice performance involves two ways: the slice provision system can admit the number of UEs for which the existing resources are sufficient; the other way is to monitor performance and react if slice overload occurs with allocation of more resources or removing UEs.

Subslicing has been shown to improve slice performance in [5] and [6]. If the slice performance depends on slice size, then subslicing a slice into suitable-sized subslices can improve the slice performance. This is applicable in close to slice overload situation when no additional bandwidth available, that is fixed-bandwidth slice or bandwidth constrained slice. In this paper we implement the monitor-analyze-plan-execute-knowledge (MAPE-K) [7] management CCL (MCCL) to introduce subslicing with the aim to improve slice performance on fixed slice bandwidth.

The contributions of this paper are as follows:

- MAPE-K CCL exists, but not for automatic subslicing. We describe it for automatic subslicing with the goal to improve RAN slice performance on fixed slice bandwidth.
- We propose a Plan function that decides based on subslice performance whether the subslice should be split, merged, or not changed. The subslice with the best size has achieved the lowest utilization, highest goodput per allocated RB, and lowest BLER.
- We propose an Execute function, where in the subslice splitting algorithm, the UEs are clustered by their achieved BLER and the slice bandwidth is allocated proportionally to the group BLER and requested sum rate. This improves the slice performance more than if the UEs are clustered by their requested rate.
- The slice is simulated and its performance is evaluated when MCCL-initiated subslicing is performed during the initialization phase and traffic increase and decrease scenarios. Slice performance improvement is shown by a reduction in slice utilization and BLER with an increase in slice goodput per RB.

The remainder of this paper is organized as follows. In Section II, the related work on management CCLs and subslicing described. The performance data at all possible subslice sizes, which is the input data for subslicing decision ML tool, is presented in Section III. The proposed MCCL is described in Section IV and evaluated in Section V. Finally, Section VI concludes the paper.

II. RELATED WORK

For automation in management, the closed control loop (CCL) operates autonomously to attain predefined objectives without human intervention. The loop consists of four functions and a shared database of knowledge. These functions collect and analyze data, make decisions and execute decided actions on the managed entity. Examples of well-known CCLs are MAPE-K (Monitor-Analyze-Plan-Execute, Knowledge) and OODA (Observe, Orient, Decide, Act) [1].

MAPE-K [7] is a management CCL used by autonomic systems for self-management. It functions in a series of steps: the Monitor function gathers performance data, the Analyze function analyzes this data, the Plan function decides changes in configuration, and the Execute function implements these changes to the configuration of the autonomic system. The Knowledge is a database containing information accessible for all other functions.

Vision of closed loop automation in [8] uses MAPE loop on O-RAN architecture for end-to-end (E2E) slice orchestration and network management. The CCL framework described in [9] contains policy-driven CCL and uses intent-based networking (IBN) to enable ZSM in a multidomain environment. It covers service management model, and presents policy generation algorithms for policy generation. [10] presents framework of collaborating MAPE-K closed loops used for end-to-end service management.

CCL types can be categorized as ready-made or made-to-order CCLs. Ready-made CCLs are pre-integrated and made-to-order CCLs are assembled on demand [1]. Our proposed MCCL can be considered as a made-to-order CCL designed to be assembled specifically for managing radio resources in the mIoT slice when a slice is in the close to overload state and no additional bandwidth resources are available.

[11] proposes AI-driven MANO system for massive slicing. They use their own CCL, which contains monitoring, analytics and decision phases, while execution is outside CCL. The monitoring and analytics components predict the RAN resources under service level agreement (SLA) constraints. The federated learning is used to improve the prediction model. The system overhead and computation load reduced, as well as slice SLA violation rate.

While we propose a CCL implementation for subslicing, the full CCL is implemented in [12] to mitigate connection loss for moving UE. UEs are ships in real-world seaport testbed. The movement of UE is predicted using deep neural network (DNN) consisting of 3 hidden layers, and radio link failure predicted. In addition, the power of the beam increased and beam down-tilted to avoid actual radio link failure.

Network slicing monitoring framework in [13] was used to measure the effect of polling interval to consumption of computing and storage resources. The slice-specific data collectors use more resources than if data collection servers used. The smaller polling interval increases resource usage, except storage resource consumption remains the same if slice-specific data collectors used. The polling intervals of 5 s and 1 s were evaluated, however, in [14] the KPIs supposed to be monitored over 30 s period.

Data analytics is discussed in [15]. For RAN the relevant analytics can be used for interference handling, channel quality prediction, control of dynamic radio topology, energy efficient improvements, multi-slice resource management enhancement. Types of analytics are: descriptive, diagnostic, predictive, and prescriptive.

The subslicing in [5] is done to group UEs with similar features into one subslice inside the slice. The purpose of subslicing was to better satisfy the requirements of UEs by clustering UEs by UE features into subslices. They evaluated clustering quality to determine the best number of subslices. They claim that slice performance in terms of throughput, power consumption and energy efficiency have been improved when slice had subslices.

CCL has been used for automatic network configuration adaptation to changes in service requirements or achieved radio signal quality. The slice performance dependence on slice BWP size is not researched. If the performance of the RAN depends on the BWP size of the RAN, the automatic change in the BWP size is not considered. The management CCL has not been used for RAN subslicing with the aim of improving slice performance under bandwidth constraints. In this paper we fill these gaps and contribute to RAN subslicing with slice performance improvement, and propose Plan and Execute functions for management CCL for automatic subslicing on fixed slice bandwidth.

III. SUBSLICE PERFORMANCE DEPENDENCE ON SUBSLICE SIZE (TRAINING DATA)

First, we discover and present the subslice performance dependence from the subslice size by simulating the subslice of all possible sizes and collecting its performance data. This is needed for decisions in which subslice sizes provide better performance and which subslices should be split or merged to improve the performance of their UEs.

The subslice works on 5G band n28 (FDD, UL 730-748 MHz, DL 758-803 MHz) [16], subcarrier spacing is 15 kHz thus maximum subslice bandwidth is 250 RBs. The band n28 can be used for both machine-to-machine (M2M) internet of things (IoT) and vehicle to everything (V2X) service verticals.

For each RB allocated to the subslice, a UE, which is able to consume that RB, is admitted to the subslice. The UE requests rates 200 kbps both in UL and in DL, and packet size used is 40 bytes. The subslice is simulated three times: if it contains all good-BLER, medium-BLER and poor-BLER UEs. To achieve the desired BLER, the UE distance from gNB should be set accordingly. BLER is assumed to increase with distance; thus, UEs that should achieve poor BLER are positioned far from the gNB. The used channel model is clustered delay line C (CDL-C), which is non line of sight (NLOS) model [18]. The smallest subslice to simulate is 4 RBs, because into smaller BWPs the sounding reference signal (SRS) necessary for channel estimation and scheduling, can not be fitted [19]. Other settings are provided in table 1.

TABLE 1. MATLAB Toolbox settings

Parameter	Value
Band	n28 (FDD, UL 730-748 MHz, DL 758-803 MHz)
Carrier frequency	UL 730 MHz, DL 758 MHz
Channel model (for both UL and DL)	CDL-C
PUSCH preparation time for UEs	200 μs
Logical channels per UE	1
RLC entity type	UM bidirectional
Duplex mode	FDD
Scheduler strategy	Round Robin
Length of scheduling cycle	1 frame
RB allocation limit UL	same as RBs for subslice
RB allocation limit DL	same as RBs for subslice
Simulation time	1 second
Subslice simulation tool from MATLAB 5G Toolbox	NR Cell Performance Evaluation with Physical Layer Integration [17] R2021b

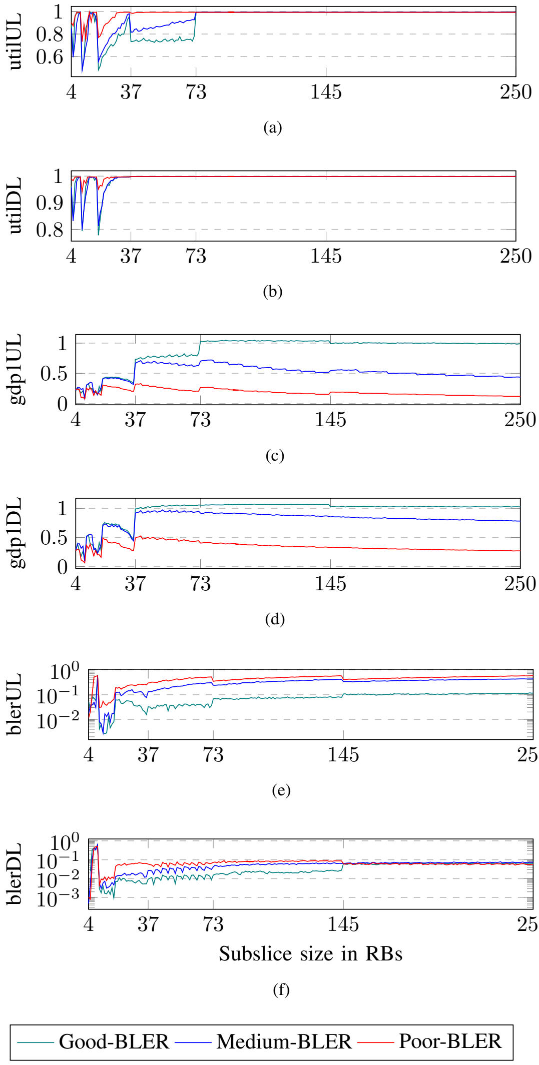
Six key performance indicators (KPI) are measured for each subslice: bandwidth utilization in UL (utilUL) and DL (utilDL), subslice goodput (application-level throughput) in Mbps per one RB in UL (gdp1UL) and DL (gdp1DL), and the subslice average block error ratio (BLER) in UL (blerUL) and DL (blerDL). The total goodput is received, but to compare the results of subslices of different size, the goodput per allocated RB is calculated. One second of working time is simulated using Matlab 5G toolbox tool called NR Cell Performance Evaluation with Physical Layer Integration [17]. Results are shown in figure 1.

In utilUL there are low peaks on size of 4, 11, and 19 RBs. In the range of 37-73 RBs the utilization is lower. Larger subslices have 100% utilization. If subslices are smaller than 37 RBs, the goodput is low. In UL there is the range of 37-73 RBs when utilUL is lower and gdp1UL is better than in smaller subslices. Subslices larger than 37 RBs, downlink utilization and goodput are both high. In UL, both utilization and goodput are high if subslice is larger than 73 RBs.

BLER is initially high, then in subslice size of 10 RBs BLER is low, because there is low goodput. Later, BLER increases with subslice size increase. For small subslices, the BLER is high because of incorrect channel estimation caused by too less reference symbols in the subband.

In UL, the subslice size range 37-73 RBs is the lowest utilUL and low BLER. The highest goodput in DL is in range of subslice size 73-145 RBs and in DL 37-145 RBs.

With medium-BLER and poor-BLER UEs the utilization was higher than with good-BLER UEs. Goodput was the highest when subslice size was smaller than when good-BLER UEs were in the subslice. Because the BLER is high, packet retransmissions are necessary, which increases



(a)

(b)

(c)

(d)

(e)

(f)

FIGURE 1. **Subslice performance data for subslice size.**

the bandwidth utilization to full, and because of bandwidth shortage, the goodput will be low. Additional bandwidth without additional UEs can improve the performance of the subslice. If no additional bandwidth is available, the selection of the best subslice size can improve the performance achieved by the UE or RB.

IV. PROPOSED MANAGEMENT CCL

A. THE LOOP

The proposed CCL is the MAPE-K type and is similar to the CCL specified in 3GPP TS 28.535 [3]. It consists of four functions. Monitor collects performance KPIs. Analyze function detects if subslicing is necessary. To obtain decisions, the optimization problem is solved to provide training data for neural network (NN). Plan (Decide) function uses this classifier NN to decide whether the subslice should be split, merged or not changed. Execute creates subslices according to planned configuration. The Knowledge is a shared database. Full CCL is shown in figure 2.

Currently, the CCL is reactive type, but it can be proactive, if values of KPIs are predicted.

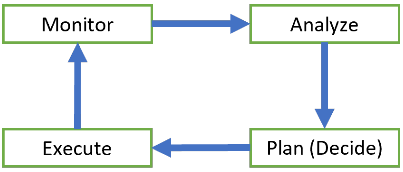


FIGURE 2. **MAPE-K-type management CCL.**

B. MONITOR

The Monitor function collects values of the six KPIs for the slice and each subslice it may have. In addition to slice bandwidth utilization, the goodput per one RB in Mbps and BLER is collected. Mean values of KPIs are available for other functions of the MCCL in the Knowledge database. Now, values are calculated for the time after execute function has changed the slice configuration.

C. ANALYZE

The poor performance of a slice or subslice is indicated by high utilization and low goodput and high BLER. Authors of [14] suggest that the high and low thresholds for bandwidth utilization can be 80% and 20%. For BLER the target value is below 0.1.

The Analyze function provides the pre-trained neural network (NN) for Decide function. Training data is the simulation results of six KPI values presented in section III. The subslice performance at different subslice sizes needs to be learned if subslice contains all good-BLER, medium-BLER and poor-BLER UEs. Then it can work better for slice which contains mixed-BLER UEs. In our previous work [20], the NN was trained using clustering result of a subslice performance data of all possible sizes of a subslice, which contained all good-BLER UEs. This approach does not work correctly in more realistic situation with mixed-BLER UEs in the slice. If poor-BLER UEs are in the slice, the slice utilization is high and goodput is low, then operation ”merge” was decided. However, our other previous work has shown that smaller subslices performed better, if UE BLER was worse [6].

To find the respective output decision (”merge”, ”no change”, ”split”), the optimization problem is solved separately for good-BLER, medium-BLER and poor-BLER simulation results.

1) Optimization problem to find best subslice size

The best size for subslice is with the lowest utilization, highest goodput per allocated RB, and lowest BLER.

Let x be the optimal size of a subslice in RBs.The relevant predictors/KPIs K for good subslice size are utilization $K^{(u)}$ in UL, $K^{(u,UL)}$ and DL $K^{(u,DL)}$, goodput per one RB $K^{(g)}$ in UL $K^{(g,UL)}$ and in DL $K^{(g,DL)}$, block error ratio (BLER) $K^{(b)}$ in UL, $K^{(b,UL)}$ and in DL $K^{(b,DL)}$.

The goal is to find a subslice size where the utilization is the lowest, goodput per one RB is the highest and BLER is the lowest. The multi-objective integer optimization problem can be formulated as:

$$\begin{cases} \min K^{(u)}(x), \\ \max K^{(g)}(x), \\ \min K^{(b)}(x). \end{cases}$$
$$\text{subject to} \begin{cases} K^{(u)}(x) \in [0,1], \\ K^{(g)}(x) > 0, \\ K^{(b)}(x) \in [0,1], \\ x \in \{4, 250\}. \end{cases} \tag{1}$$

This optimization problem needs to be solved for each dataset separately for subslices containing good-BLER, medium-BLER and poor-BLER UEs.

Instead of using multiple functions, the objective function can be formulated as the sum of differences between KPI values and the best (ideal) KPI values of the dataset. The best values of KPIs are calculated for each of 3 datasets separately as follows

$$\begin{aligned} K^{(u,best)} &= \min_{i=4,\ldots,250} K_i^{(u)}, \\ K^{(g,best)} &= \max_{i=4,\ldots,250} K_i^{(g)}, \\ K^{(b,best)} &= \min_{i=4,\ldots,250} K_i^{(b)}. \end{aligned} \tag{2}$$

The difference, ΔK, for each subslice size can be achieved by calculation of root-mean-square error (RMSE) between the current and best values of KPIs using the following formula:

$$\Delta K^{(j)} = \sqrt{(K_i^{(j)} - K^{(j,best)})^2}, i = 4, ..., 250, j = 1, ..., m. \tag{3}$$

The objective function is to minimize the sum of RMSEs of utilization, goodput and BLER from the best values of utilization, goodput and BLER.

Objective function can be written as:

$$\min_{N^{(RB)}} \sum_{j=1}^{m} \Delta K^{(j)}, \tag{4}$$

where the number of KPIs is $m = 6$.

The objective value a for each subslice i is calculated

$$a_i = \sum_{j=1}^{m} \Delta K^{(j)}. \tag{5}$$

The optimization problem is solved for subslice size performance data presented in section III. The objective values for each subslice size are calculated using equation 5 and results are plotted in figure 3.

The size of a subslice with minimum objective value is the best size for a subslice. Smaller subslices will have decision “merge”. Other subslices with a little greater objective value could also be in a good size and have the decision “no change“. The ε-neighbourhood is considered as percentage from the range of objective value shown on the vertical axis of the plots in figure 3. The “merge” zone is on the left side until the subslice size with the minimum objective value, which is shown in the horizontal axis. The “split” zone has bounds on horizontal axis starting the best subslice size until maximum subslice size, if objective values are higher than a “tolerance” threshold denoted by ε-neighbourhood. The good-BLER subslices have larger ”merge” area, because the best subslice size is 52 RBs (3a), while the poor-BLER subslices have subslices to be merged at sizes below 10 RBs (3c). If ε-neighbourhood is larger, then more subslices are in good size and can have “no change” decisions. Thus, less subslices needed to split. More “no change“ decisions mean less reconfiguration, however it may miss better performance which can be achieved by splitting or merging of subslices.

This formulation of the objective function and calculation of the objective value enables to map the decisions to subslice performance if the subslice size vs subslice performance is different from our dataset. The subslices which have small objective values calculated from their KPIs, have good performance and their size is suitable. The objective value was calculated as a RMSE between actual and best values of KPIs of the dataset which contained the UEs in the same BLER group.

The training data for NN is somewhat different for both ε values and output decisions for subslices at all sizes and all BLER groups are shown in figure 4. NN will get the values of six KPIs of the subslice sizes as the inputs and the output decision for the subslice sizes.

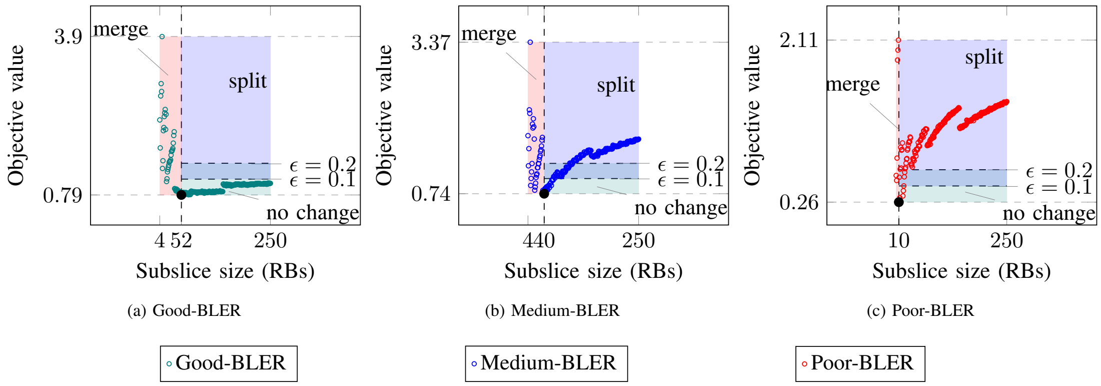
D. PLAN (DECIDE)

The “Decide” function in the MCCL cycle contains a classifier neural network (NN) which decides the slice operation, whether it should be merged, split or not changed. The training data is the simulation results, 6 KPIs, with the output class determined by solving the optimization problems for each BLER group.

The NN is trained using three datasets which have good-BLER, medium-BLER and poor-BLER UEs in the subslice. The NN inputs are values of six KPIs and not the subslice sizes nor UE BLERs. Then we assume that the NN is trained for the decisions, if the slice contains UEs with any BLER.

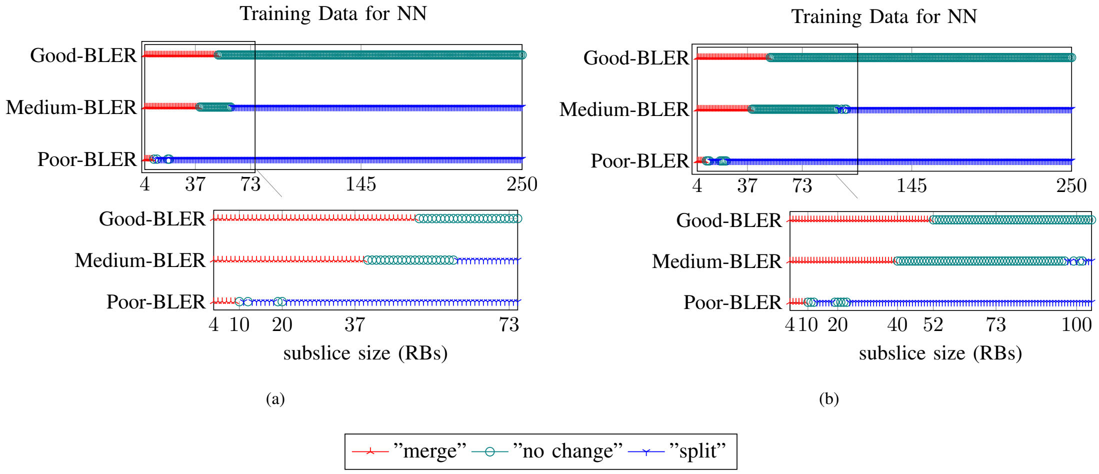
Classifier NN is selected because it can learn any input training data NN settings determined by trials. NN-6-9-3 was selected: 6 inputs (6 KPIs), one hidden layer consisting of 9 neurons, and 3 outputs (one for each decision)

The NN settings of hidden layers are determined by trials. The NNs with 1-2 hidden layers and 2–22 neurons on layer



(a) Good-BLER (b) Medium-BLER (c) Poor-BLER

FIGURE 3. Solution of the optimization problem to find best subslice sizes.



(a) (b)

FIGURE 4. Training data output decisions for NN with (a) $\varepsilon = 0.1$ and (b) $\varepsilon = 0.2$ is used for deciding the change of subslice configuration.

are trained, validated and tested. Then the cross entropy loss is calculated. The least number of layers and neurons where the losses are not decreasing further, is one hidden layer containing 9 neurons, thus the settings are NN-6-9-3. The NN is shown in figure 5.

Each subslice is decided one-by-one by classifier NN. The minimum subslice size is determined to be the best value of the subslice size, and it must not be greater than a half of the bandwidth part of a subslice to be split.

E. EXECUTE

Execute has subslice configuration data: UE data (UE IDs, requested rate and achieved BLER) and subslice data (number of RBs, number of UEs and UE IDs in the subslice) and decided operation for each subslice. The execute algorithm is Alg. 1. First, subslices with decision "no change" are not changed.

Secondly, subslices with decision "merge" are merged: the smallest is merged with the largest and until all subslices are merged pair-wise. This can avoid merged subslices being too large. If there is an odd number of subslices then 3 smallest are merged. The merged subslices should not become too large, because too large subslice exhibits poor performance again. If just one to merge, then its configuration will not change. This will not change the subslices which should not be changed. Subslice merging combines sets of UEs of subslices to be merged, and combines the number of RBs of subslices to be merged.

Thirdly, subslices with decision "split" are split: UEs are clustered into two and slice bandwidth is split into two, that is RBs are divided to UE groups proportional to group

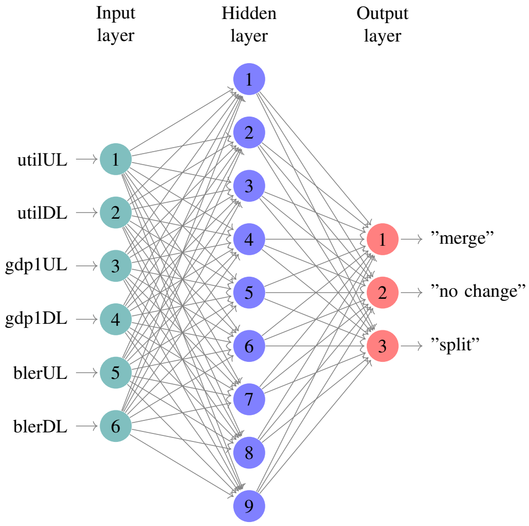


FIGURE 5. Trained NN.

Algorithm 1 Execute function

```
Sub 1 ▷ Process subslices with a decision "no change"
The subslice UEs, RBs and $id$ is not changed.
Sub 2 ▷ Process subslices with a decision "merge"
Order subslices ascending
$n \leftarrow$ number of subslices to be merged
if $n \leq 3$ then
    Merge all subslices
else if $n$ is odd then
    Merge 3 smallest subslices
else
    repeat
        Merge the largest subslice with the smallest subslice
    until Done
end if
Sub 3 ▷ Process subslices with a decision "split ($S_{min}$)"
for Each subslice do
    Split subslice into 2 with minimum subslice size constraint
end for
```

Algorithm 2 Split function

```
Convert $S_{min}$ to $R_{min}$
Sub 1 ▷ Cluster UEs: if clustering algorithm is DSbR then by UE Rate, if DSbB then by UE BLER
repeat
    Modified k-means
until All clusters' sum rate$\geq R_{min}$
repeat
    Original k-means
until All UEs assigned to a cluster
Sub 2 ▷ Allocate slice RBs to UE groups
Initial allocation proportional to group sum rate
Second allocation proportional to group BLER
Third allocation
while leftover RBs do
    Allocate leftover RB to the group consisting UE with highest BLER, descending order
end while
```

requested sum rate and group BLER. The subslice splitting algorithm is Alg. 2.

The base algorithm for splitting a subslice is taken from [6] and modified to fit to the UEs which request different rates. First, the minimum subslice size in RBs, S_{min} is converted to minimum subslice sum rate using

$$R_{min} = \left\lfloor S_{min} \cdot \frac{R^{(s)}}{N^{(RB)}} \right\rfloor, \tag{6}$$

where $R^{(s)}$ is the slice sum rate, $N^{(RB)}$ is the number of slice RBs.

The modified k-means stops when the requested minimum sum rate is reached. Sum rate is calculated as a sum of requested rates of the UEs in the cluster. Minimum sum rate is calculated from the sum rate of UEs of the subslice to be split and number of subslice RBs.

V. PERFORMANCE EVALUATION

The aim is to investigate how the proposed MCCL works and whether the slice performance is improved if subslicing is done.

A. SIMULATION SETUP

Slice bandwidth is band n28, 45 MHz (250 RBs, subcarrier spacing is 15 kHz). Slice has a set of 250 UEs which use 40-byte packet sizes. The requested rates are 100, 200 or 400 kbps, symmetrical for UL and DL. The slice sum rate is 50 Mbps. With these UE settings, the slice SLA matches the SLA of 250-RB subslice in section III by number of UEs and requested sum rate. To have the same number of UEs in the slice, and the same requested sum rate of the slice, some UEs request 400 kbps to consume 2 RBs, and some RBs are consumed by 2 UEs, each requesting 100 kbps. The UE distances are such that one third can achieve good BLER, one third can achieve medium BLER and one third can achieve poor BLER. The UE parameters are shown in table 2.

Slice is simulated using MATLAB 5G Toolbox tool called NR Cell Performance Evaluation with Physical Layer Integration [17] R2021b. Settings are shown in table 1. The slice is simulated 10 times in each setting.

TABLE 2. **Set of 250 UEs**

BLER	Requested rate	Number of UEs
Any	100 kbps	110
Any	200 kbps	85
Any	400 kbps	55
Total	**50 Mbps**	**250**
Good-BLER	Any	83
Medium-BLER	Any	84
Poor-BLER	Any	83
Total	**50 Mbps**	**250**
Good-BLER	100 kbps	36
Good-BLER	200 kbps	28
Good-BLER	400 kbps	19
Medium-BLER	100 kbps	37
Medium-BLER	200 kbps	29
Medium-BLER	400 kbps	18
Poor-BLER	100 kbps	37
Poor-BLER	200 kbps	28
Poor-BLER	400 kbps	18
Total	**50 Mbps**	**250**

The slice performance is compared if the MCCL uses following subslice splitting algorithms:

- "static 3 subslices" (S3S) is proposed in [5], where subslices are created based on UE features. Our set of UEs has 3 different rate requirements, thus the slice is always subsliced into three;
- "dynamic subslicing with UE clustering by requested rate" (DSbR);
- "dynamic subslicing with UE clustering by UE BLER" (DSbB);
- no subslicing, i.e. slice is not split.

Subslice splitting algorithms contain RB allocation proportional to the group sum rate and group BLER.

B. RESULTS OF INITIALIZATION

During the subslicing initialization, MCCL is run as many times as it takes to reach a stable configuration of subslices in the slice, a convergence, that is, when all decisions are "no change".

The initialization of algorithm S3S contains just one iteration to group UEs by requested rate and allocate RBs proportional to group sum rate and group BLER. Slice configuration is shown in the figure 6. For dynamic subslicing algorithms, the MCCL is run and subslicing is done according to thedecisions provided by NN based on six KPIs of the slice/subslice.

The training data for the NN were obtained from the optimization problem. There are different options for mapping the decision to the subslice size based on its objective value. The horizontal boundary between decision "no change" and "split" depends on the selected size of ε-neighbourhood, see figure 3. Two values of ε, 0.1 and 0.2 are selected for subslicing initialization for comparison. The training data for NN of the selected ε values are shown in 4 (a) and (b), respectively.

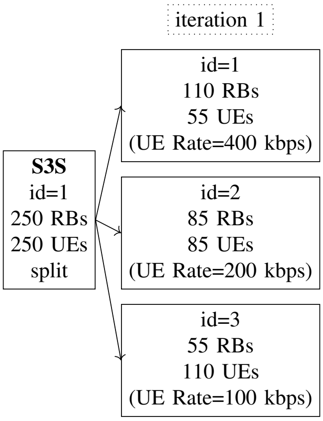


FIGURE 6. **Initialization of subslicing algorithm S3S [5] (3 subslices, each for different UE rate).**

Next, the slice configuration is compared if dynamic subslicing algorithms were used for initialization. The slice configuration changes during initialization with DSbR splitting algorithm is shown in figure 7.

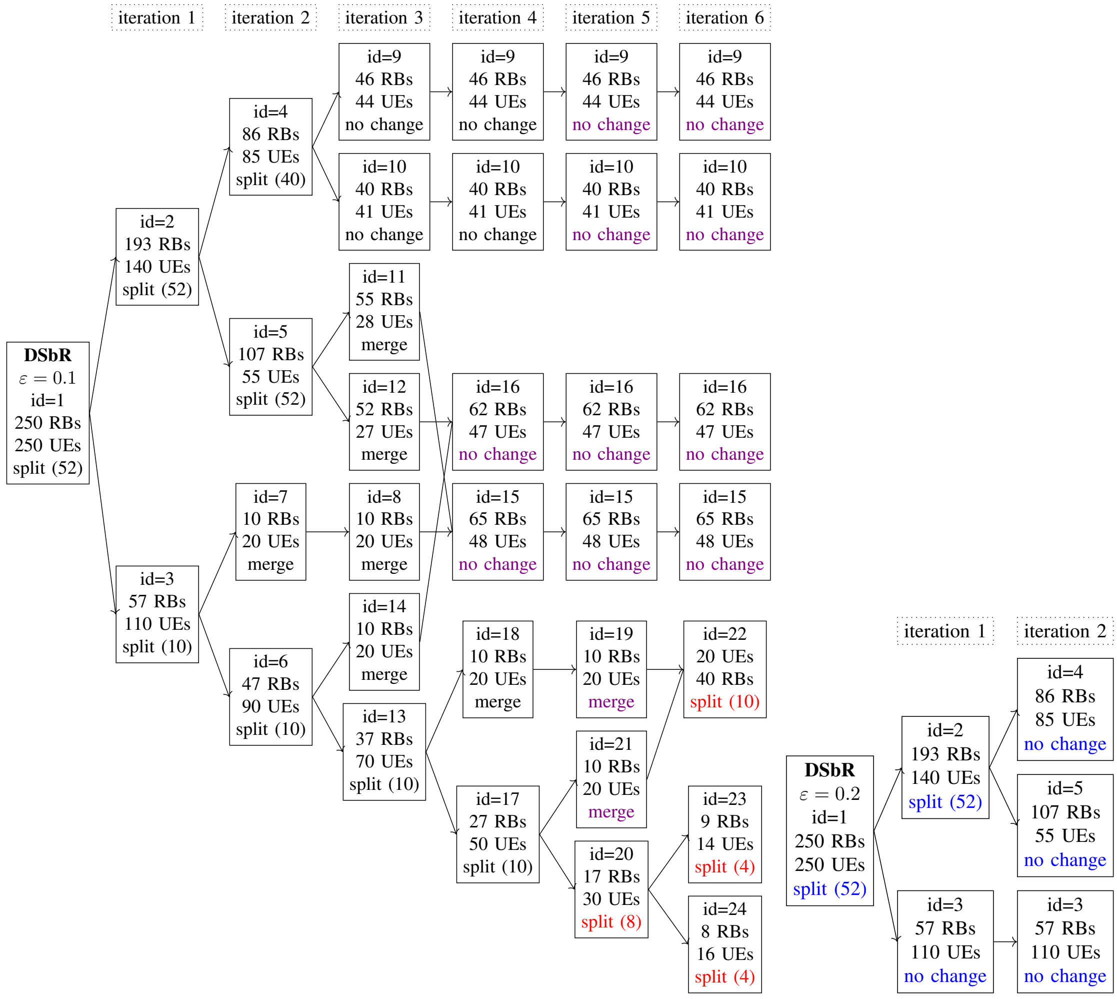
If $\varepsilon = 0.1$ the initialization using DSbR took six iterations (see figure 7a). In the third iteration, it was decided two pairs of subslices to be merged, and this resulted good subslices with $id = 15$ and $id = 16$. Subslice with $id = 13$ was decided to split on iteration 3. Of its subslices in iteration 4, one was decided to be merged, and one was decided to be split. Finally, some subslices decided to be split into sizes of four or five RBs. With greater ε and splitting algorithm DSbR the stable slice configuration was reached faster, within just two iterations, as shown in figure 7b.

The algorithm DSbB needed 4 iterations with both ε values, as shown in figure 8. However, in both values of ε, there is a subslice with split-merge infinite loop in configuration change. The infinite loop is that in one iteration the subslice should be split, and in the next iteration both subslices should be merged.

The initialization of the slice configuration required fewer iterations, and the slice contained fewer subslices with a greater value of ε. If MCCL uses the splitting algorithm DSbR, then the slice contains three subslices, as if the splitting algorithm is S3S. When the splitting algorithm DSbB was used, a split-merge infinite loop appeared for one subslice.

The slice performance results during the initialization are shown in figure 9. Left column contains slice KPI values if $\varepsilon = 0.1$ and right column contains slice KPI values if $\varepsilon = 0.2$.

The static algorithm S3S achieves better slice performance than no subslicing: UL utilization has improved by 1.26%,



(a) (b)

FIGURE 7. **Initialization of scenario DSbR completed. (a)** $\varepsilon = 0.1$ **and (b)** $\varepsilon = 0.2$.

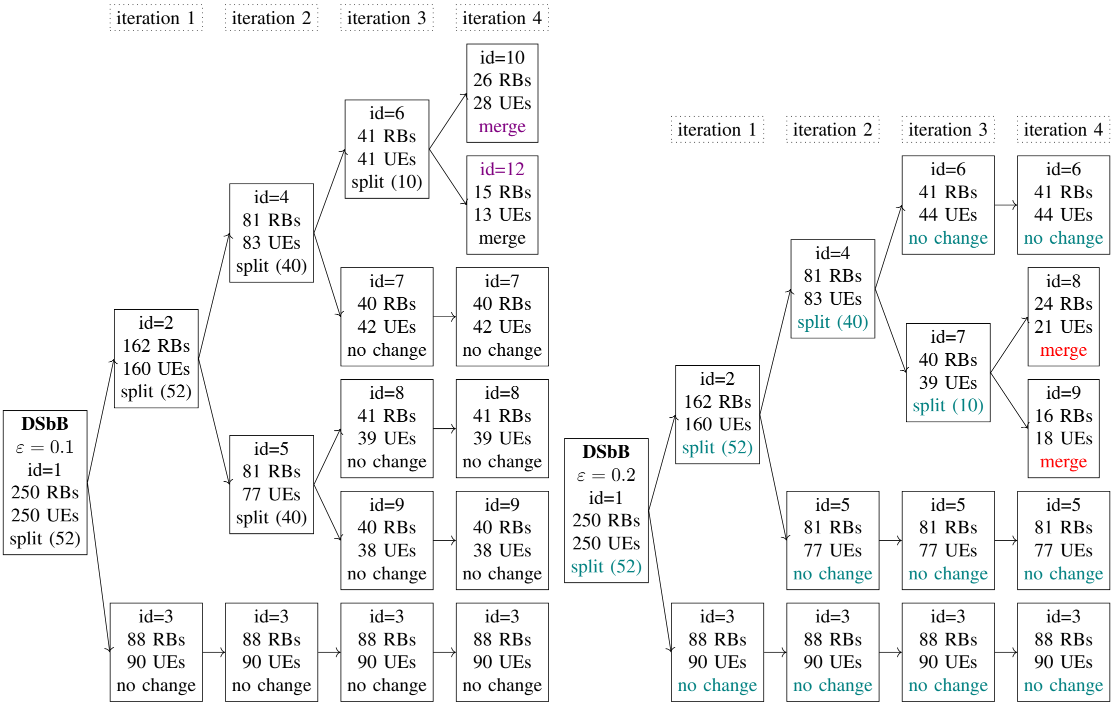
goodput per one RB improved in UL by 28.4% and in DL by 11.5%, and BLER reduced by 31% and 43.7% in UL and DL, respectively.

Dynamic splitting algorithms achieve different performance if different ε values were used. The increase of ε improved slice goodput and BLER for DSbR, however with DSbB the change of KPI values was small. However, despite the split-merge infinite loop, DSbB achieved the best slice performance improvement in reducing slice utilization by 6% in UL, BLER by about 60% in both UL and DL, and improved goodput per one RB in UL the most, 37%. The DSbB had better goodput in iteration 3, but after subslice splitting, the achieved gootput decreased. The dynamic algorithms can miss their peak goodputs if the decisions contain too much splitting of the subslices.

For runtime simulations, the dynamic subslicing algorithms were initialized using training data obtained with $\varepsilon = 0.2$ because initialization requires fewer iterations and slice performance is generally better.

C. RUNTIME SCENARIOS

Two scenarios were used to investigate how the proposed MCCL works in the conditions when slice load increases or decreases. It is expected that subslicing can improve the slice performance on a fixed slice bandwidth. The starting point of the scenarios is the initialized slice from the previous


iteration 1
iteration 2
iteration 3
iteration 4
id=10 26 RBs 28 UEs merge
id=6 41 RBs 41 UEs split (10)
id=12 15 RBs 13 UEs merge
id=4 81 RBs 83 UEs split (40)
id=7 40 RBs 42 UEs no change
id=7 40 RBs 42 UEs no change
id=2 162 RBs 160 UEs split (52)
id=8 41 RBs 39 UEs no change
id=8 41 RBs 39 UEs no change
DSbB $\varepsilon = 0.1$ id=1 250 RBs 250 UEs split (52)
id=5 81 RBs 77 UEs split (40)
id=9 40 RBs 38 UEs no change
id=9 40 RBs 38 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
iteration 1
iteration 2
iteration 3
iteration 4
id=6 41 RBs 44 UEs no change
id=6 41 RBs 44 UEs no change
id=4 81 RBs 83 UEs split (40)
id=8 24 RBs 21 UEs merge
id=7 40 RBs 39 UEs split (10)
id=2 162 RBs 160 UEs split (52)
id=9 16 RBs 18 UEs merge
DSbB $\varepsilon = 0.2$ id=1 250 RBs 250 UEs split (52)
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=5 81 RBs 77 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change

(a) (b)

FIGURE 8. Initialization of scenario DSbB completed. (a) $\varepsilon = 0.1$ and (b) $\varepsilon = 0.2$.

subsection. The slice load increase is performed in Scenario 1, in which a new UE comes into the subslice with the lowest utilization. The slice load decrease is performed in Scenario 2, where one UE with the highest BLER leaves. The UE that is coming or leaving has high requested rates and is expected to achieve a high (poor) BLER.

The discrete-event simulator shown in figure 10 is created to run the scenarios. Initially, in round 0, the slice has 250 UEs, and the initial configuration of the subslices is based on the splitting algorithms used in the previous subsection. At the end of the round, an event occurs, which means adding or removing one UE, as stated in the scenario. The next round begins with the simulation of a slice to determine the effect of the event. The MCCL then runs and possibly changes the slice configuration for subslicing. The second simulation of the slice in this round shows the effect of the new configuration on slice performance.

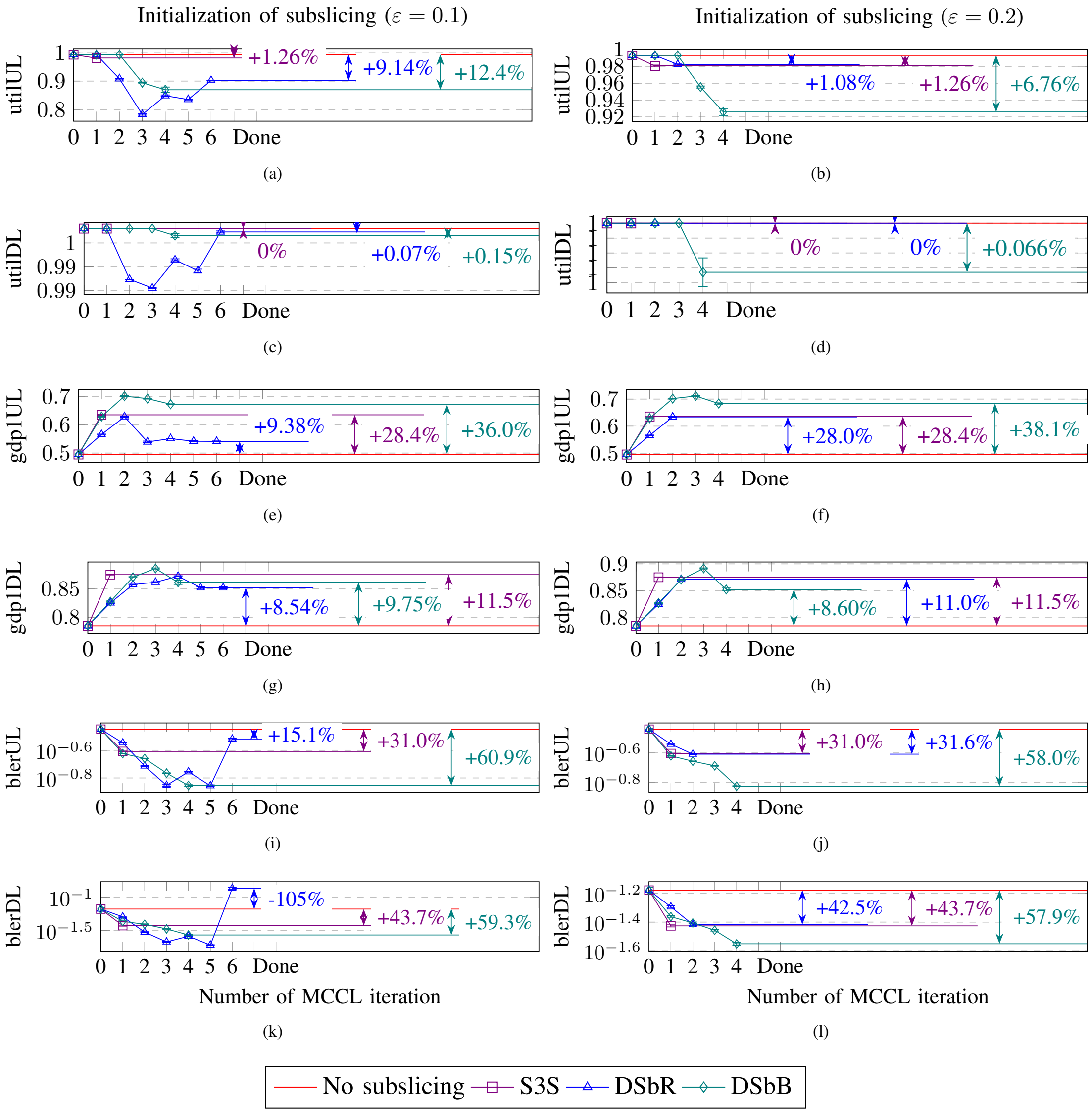
D. RUNTIME RESULTS

The scenarios run seven events and each slice was simulated 10 times. The slice performance changes are shown in figure 11. The runtime simulation results are shown in figure 13.

1) Scenario 1

Let us consider the slice performance data of scenario 1 (figure 11a and 11c). Within 7 events, the slice requested sum rate increases by 5.6% due to new UEs added. When no subslicing done, the slice utilization did not change, goodput per one RB decreased by 2% and 1% in UL and DL, respectively, and BLER increased by 1% in UL and decreased by 2% in DL. If subslice splitting by UE requested rates (algorithms S3S and DSbR) were used, then utilization in UL increased, goodput decreased and BLER increased a few more percent. When subslice splitting by UE BLER (algorithm DSbB) was used, then the change of performance depends on how many subslices are in the slice. After the initialization, the slice contained 5 subslices. After events 5, 6 and 7, the slice contained 4, 5 and 6 subslices, respectively (see table 12. When less subslices, then UL utilization and UL goodput per one RB increased more, up to 5%. In DL, the utilization did not change and goodput increased 4% if 4 subslices, and decreased 2% if 6 subslices. The greatest effect of number of subslices had BLER in both UL and DL. The BLER increased up to 28% if 4 subslices, and decreased up to 41% if 6 subslices were in the slice.

Details on figure 13 left column show dynamics of KPI value changes during the events and MCCL runs. If no



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

FIGURE 9. **Slice performance during initialization of subslicing algorithms (a), (c), (e), (g), (i), (k) $\varepsilon = 0.1$ and (b), (d), (f), (h), (j), (l) $\varepsilon = 0.2$. Percentages show how much better values were achieved compared to those without subslicing.**

subslicing the utilization is full already. The UL utilization increases to full utilization, if splitting algorithms S3S and DSbR were used. If DSbB then utilization is lower than others. The utilization in DL does not change, being full. In a few events after MCCL run, the DSbB can achieve slightly lower utilization in DL. The goodput per one RB in UL slowly decreases, but DSbB achieves best values. In DL, the values of goodput per one RB for DSbB changes, being sometimes better and sometimes worse than S3S and DSbR. Splitting algorithms S3S and DSbR can achieve slice BLER similar and not changing. If DSbB splitting algorithm used, the slice BLER values depend on the number of subslices, but still BLER is the smallest in UL and DL.

The slice configurations are shown in figures 14 for SCS, figure 15 for DSbR and figure 16 for DSbB splitting algorithm used in MCCL, respectively.

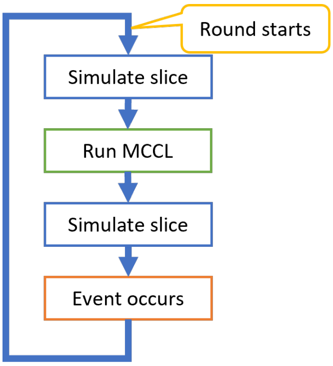


FIGURE 10. **Discrete event simulator.**

The slice configuration, when S3S was used as splitting algorithm in MCCL, had always 3 subslices, each for UEs with different requested rate. UEs are coming in to the largest subslice and resource allocation increases the number of RBs for this subslice by taking RBs away from other subslices. When DSbR was used as splitting algorithm, then slice configuration is stable, consisting of 3 subslices, similarly each for UEs with different requested rate. New UEs come to the subslice with $id = 5$ because this had the lowest bandwidth utilization. The 7th UE comes to the subslice with $id = 3$ because now this had the lowest utilization. The subslice splitting algorithm DSbB had created larger stable subslices for UEs achieving good or medium BLER. The two subslices with $id = 8$ and $id = 9$ contain UEs with poor BLER and are split or merged at each MCCL run. When the 4th and 7th UE added, then there were 6 subslices in the slice, and both goodput and BLER were low. When the slice contained 4 subslices then the goodput achieved the highest values and so did BLER. The best goodput per RB in UL and DL was achieved using splitting algorithm DSbB and 4 subslices.

When slice load increases, the achieved goodput will not increase, however subslice splitting has saved on the initialization few RBs, which will be utilized and a small increase in utilization and goodput is achieved quickly. BLER is lower if poor-BLER UEs are divided into more subslices and this can be done by using subslice splitting algorithm DSbB.

2) Scenario 2

Let us consider slice performance data of scenario 2 (see figures 11b and 11d. Within 7 events, the slice requested sum rate decreases by 5.6% due to UEs removed. When no subslicing used, then after 7 events the slice utilization, goodput and BLER did not change more than 1%, and slice BLER in DL decreased by 4%. Subslice splitting by UE BLER (DSbB), the values of KPIs depend on how many subslices the slice contained. After the initialization, the slice contained 5 subslices. After events 5, 6 and 7, the slice contained 5, 6 and 4 subslices, respectively (see table 12. If less subslices then all KPIs were increased compared to the slice configuration containing 5 subslices at the initialization. The BLER had the greatest change. If 6 subslices then the BLER decreased by more than 40%, while utilization decreased by 7% and 1% in UL and DL, respectively.

After 7 events, the slice utilization in UL decreases by 3-4% if subslice splitting algorithms were used, as shown in figure 11b. It did not affect slice utilization in DL, which did not change. Details in figure 13 right column show that the slice bandwidth was fully utilized, and after 7 events the slice utilization was still full. With subslicing the utilization in UL slowly decreases, but it decreases the most when splitting algorithm DSbB was used. In UL, the best goodput per RB is achieved with splitting algorithm DSbB. In DL, the goodput per RB of DSbB is sometimes the best and sometimes just better than if no subslicing. The subslicing decreases the slice BLER in both UL and DL compared to no subslicing, but the splitting algorithm DSbB achieves the lowest BLER regardless of the number of subslices.

The slice configurations are shown in figure 17 for SCS, figure 18 for DSbR and figure 19 for DSbB splitting algorithm used in MCCL, respectively.

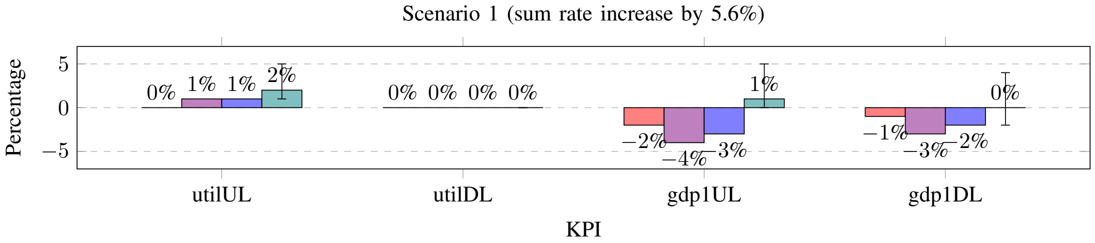
The slice configuration by using S3S splitting algorithm has always 3 subslices where UEs are clustered by requested rate. UE leaves the subslice with $id = 1$.

The scenario with no subslicing has all KPIs unchanged, however with subslicing the goodput per one RB increases and BLER decreases. Resource allocation takes RBs from that subslice and gives those to the others. Similarly, the slice contains 3 subslices by UE rate, if splitting algorithm DSbR was used. The UE always leaves the subslice with $id = 5$ because this contained UEs which request 400 kbps. The subslice splitting algorithm DSbB has stable subslices for good-BLER and medium-BLER UEs. The 2 subslices with $id = 8$ and $id = 9$ are for poor-BLER UEs and their configuration changes on each MCCL run. Similarly, the best goodput per one RB is achieved if these poor-BLER subslices are merged to one, as in events 2, 4 and 7.

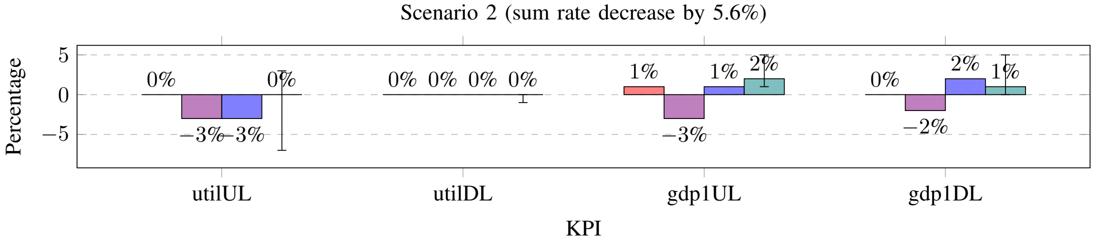
When slice load decreases then the goodput can increase because there are more resources available for packets which needed retransmission before. Similarly, as in slice load increase, the poor-BLER UEs need more subslices and the slice BLER can be decreased significantly.

VI. CONCLUSION

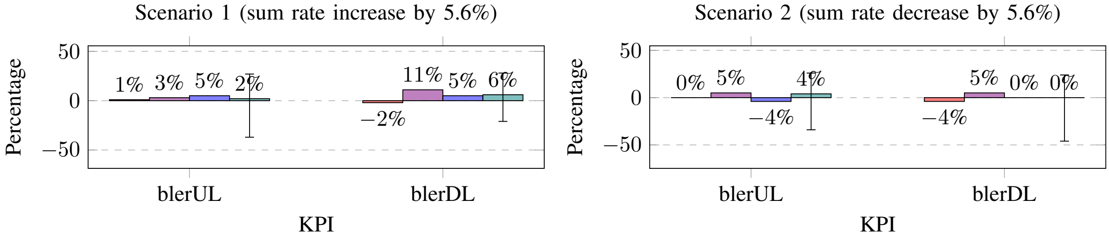
The management closed control loop for subslicing to improve slice performance was implemented in this paper. For slice simulations, the realistic BWP, n28, was used as the fixed slice bandwidth of 50 MHz. The realistic set of 250 V2X or MIoT UEs are set to consume the allocated bandwidth by their requested sum rate of 50 Mbps. The planned slice load was close to slice overload. UEs were placed at different distances from gNB to have UEs with different



(a) Scenario 1, utilization and goodput per one RB



(b) Scenario 2, utilization and goodput per one RB



(c) Scenario 1, BLER

(d) Scenario 2, BLER

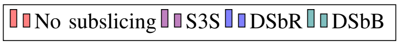
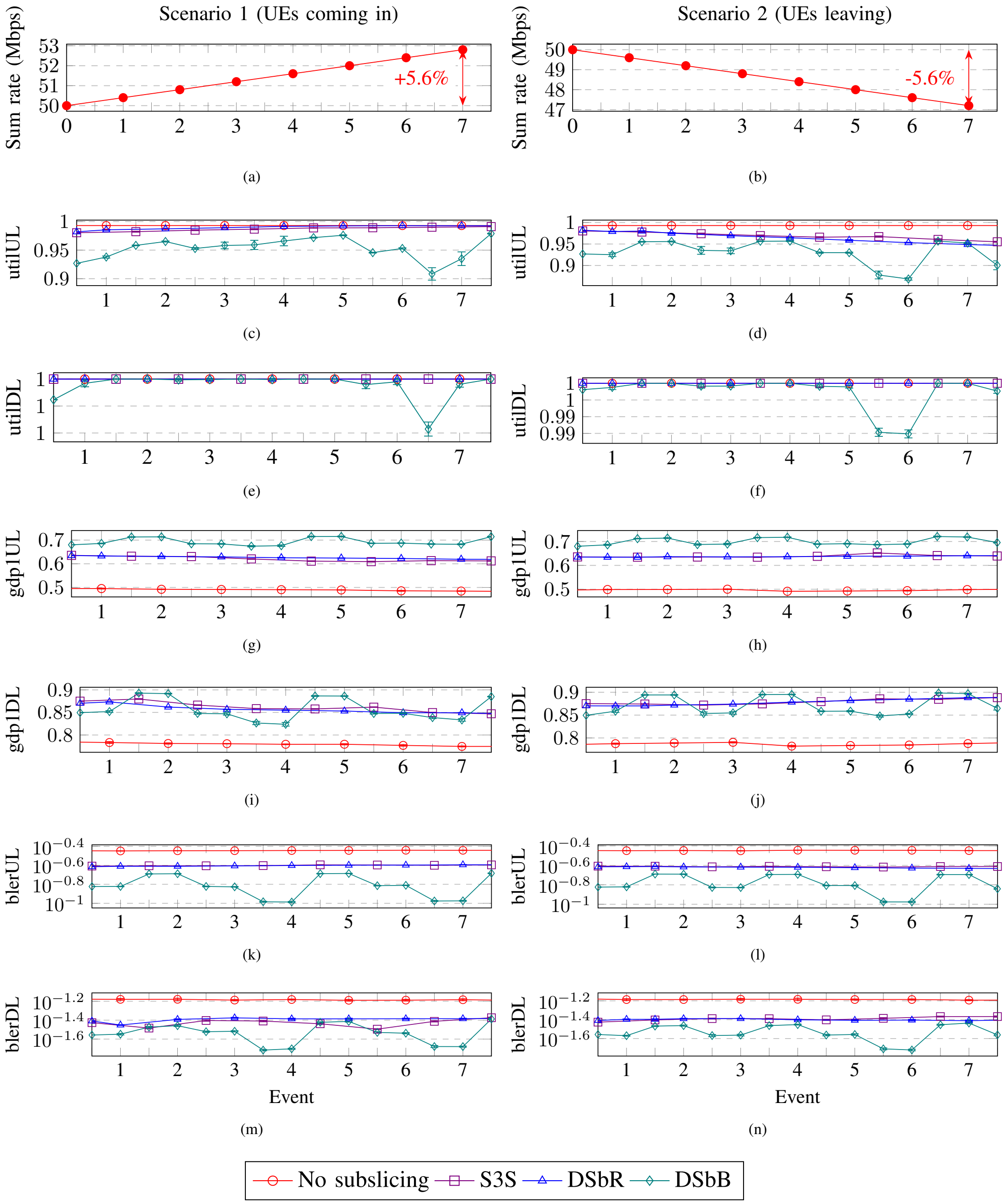


FIGURE 11. **The KPI value changes after 7 events of runtime scenarios. Percentages show difference between KPI value of the initialization and the 7th event if the splitting algorithm was used in runtime scenarios. In DL, the utilization did not change, being full.**

Scenario 1 (sum rate increase)							
Number of subslices	**Event number**	**utilUL**	**utilDL**	**gdp1UL**	**gdp1DL**	**blerUL**	**blerDL**
4	5	5%	0%	5%	4%	27%	28%
5	6	2%	0%	1%	0%	2%	6%
6	7	1%	0%	0%	-2%	-41%	-33%
Scenario 2 (sum rate decrease)							
Number of subslices	**Event number**	**utilUL**	**utilDL**	**gdp1UL**	**gdp1DL**	**blerUL**	**blerDL**
4	7	3%	0%	5%	5%	26%	25%
5	5	0%	0%	2%	1%	4%	0%
6	6	-7%	-1%	1%	0%	-42%	-46%

FIGURE 12. **The change of slice performance change if the splitting algorithm was DSbB. After initialization, the slice contained 5 subslices. After events number 5, 6 or 7, the slice contained 4, 5 or 6 subslices.**



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m) (n)

FIGURE 13. **Slice performance on events of scenarios and effect of MCCL.**

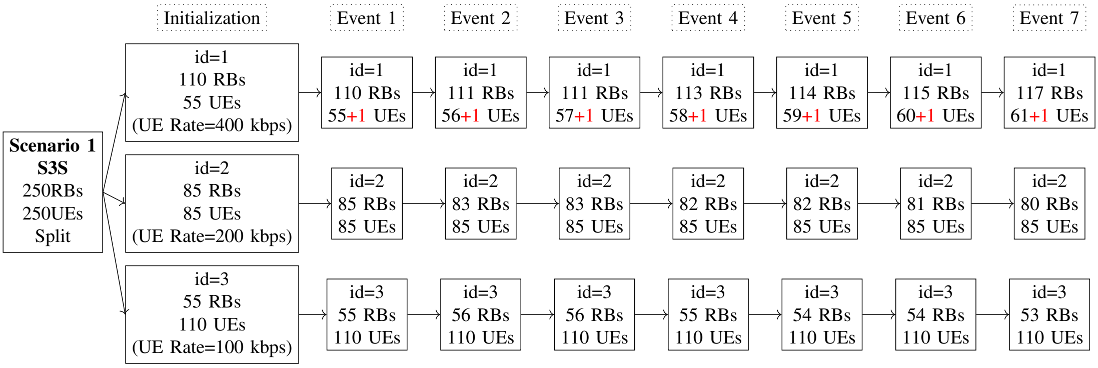


FIGURE 14. Slice configuration, scenario 1, splitting algorithm S3S, adopted from [5], first 7 events.

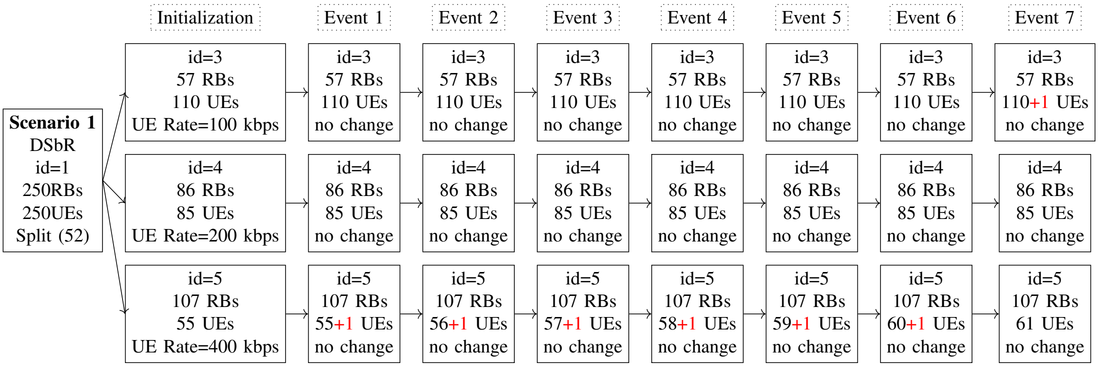

Initialization
Event 1
Event 2
Event 3
Event 4
Event 5
Event 6
Event 7
Scenario 1 DSbR id=1 250RBs 250UEs Split (52)
id=3 57 RBs 110 UEs UE Rate=100 kbps
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110+1 UEs no change
id=4 86 RBs 85 UEs UE Rate=200 kbps
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=5 107 RBs 55 UEs UE Rate=400 kbps
id=5 107 RBs 55+1 UEs no change
id=5 107 RBs 56+1 UEs no change
id=5 107 RBs 57+1 UEs no change
id=5 107 RBs 58+1 UEs no change
id=5 107 RBs 59+1 UEs no change
id=5 107 RBs 60+1 UEs no change
id=5 107 RBs 61 UEs no change

FIGURE 15. Slice configuration, scenario 1, splitting algorithm DSbR, first 7 events.

BLERs in the slice. Actually, UEs with mixed BLERs cause bandwidth overutilization due to packet retransmissions. The MAPE-K management CCL was implemented to split or merge subslices if it can improve slice performance on fixed slice bandwidth. Using the discrete event simulator, it is possible to show the dynamics of slice performance and the effect of subslicing.

Slice performance was compared if different subslice splitting algorithms were used in management CCL. For the sake of slice performance, UEs with the similar BLER in one subslice achieve better slice performance than UEs with similar requested rates.

The subslicing enables to increase goodput about 30% in UL and 10% in DL, reduce BLER at least 30% and 40% in UL and DL, respectively. Dynamic subslicing with UE clustering by BLER enables to achieve up to 6% less utilization and up to 10% more goodput, and reduce BLER additional at least 20% compared to other subslice splitting algorithms tried in the implemented management closed control loop.

In runtime scenarios when slice sum rate was increased or decreased, the subslice splitting algorithms where UEs were clustered by UE rate (S3S and DSbR), the slice configuration was stable. If subslice splitting algorithm where UEs were clustered by UE BLER (DSbB) was used, the slice configuration changed on each MCCL cycle for poor-BLER subslices. However, the slice achieved higher goodput per one RB and lower BLER than with other subslice splitting algorithms. For slice utilization and BLER improvement, the splitting algorithm should create more subslices for poor-BLER UEs, and for goodput improvement less subslices for poor-BLER UEs. Slice performance results have shown that BLER is the most sensitive KPI, dependent on the number of poor-BLER subslices. The more slice performance improvement was achieved for UL.

The future work is to find the slice performance thresholds when to start and stop perform subslicing to improve the slice performance on fixed slice bandwidth. Other research direction is to investigate how to decide the subslicing if the

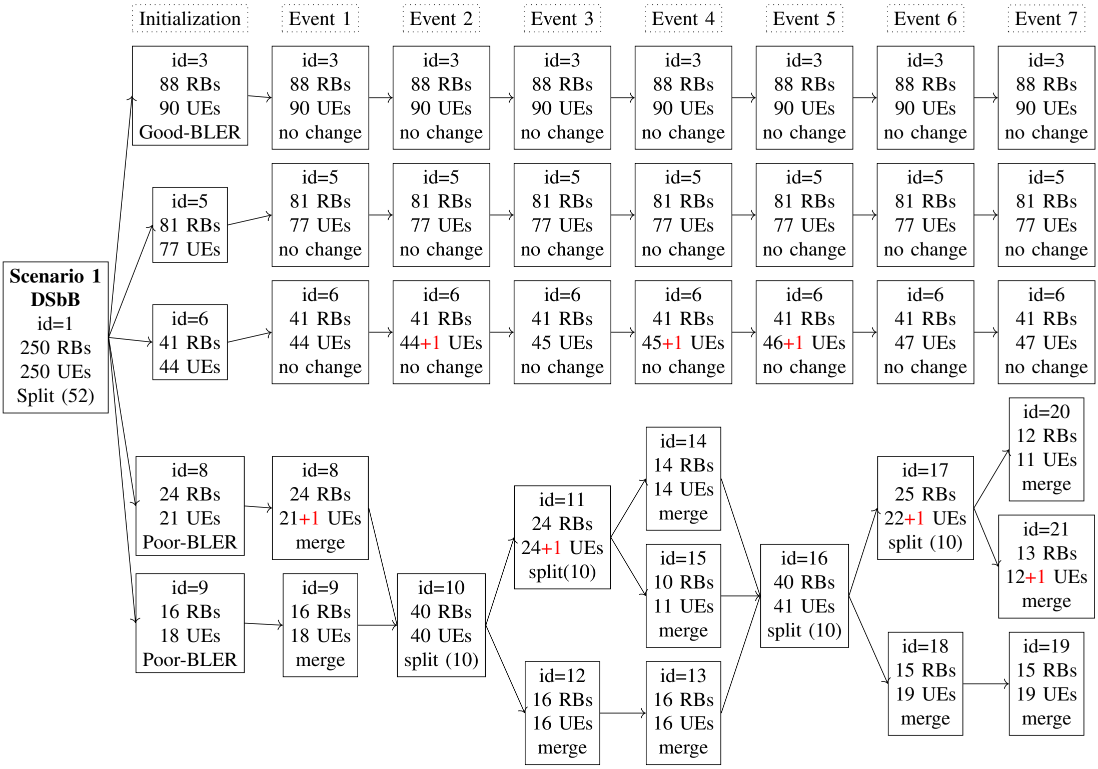

Initialization
Event 1
Event 2
Event 3
Event 4
Event 5
Event 6
Event 7
Scenario 1
DSbB
id=1
250 RBs
250 UEs
Split (52)
id=3 88 RBs 90 UEs Good-BLER
id=3 88 RBs 90 UEs no change
id=5 81 RBs 77 UEs
id=5 81 RBs 77 UEs no change
id=6 41 RBs 44 UEs
id=6 41 RBs 44 UEs no change
id=6 41 RBs 44+1 UEs no change
id=6 41 RBs 45 UEs no change
id=6 41 RBs 45+1 UEs no change
id=6 41 RBs 46+1 UEs no change
id=6 41 RBs 47 UEs no change
id=8 24 RBs 21 UEs Poor-BLER
id=8 24 RBs 21+1 UEs merge
id=9 16 RBs 18 UEs Poor-BLER
id=9 16 RBs 18 UEs merge
id=10 40 RBs 40 UEs split (10)
id=11 24 RBs 24+1 UEs split(10)
id=12 16 RBs 16 UEs merge
id=14 14 RBs 14 UEs merge
id=15 10 RBs 11 UEs merge
id=13 16 RBs 16 UEs merge
id=16 40 RBs 41 UEs split (10)
id=17 25 RBs 22+1 UEs split (10)
id=18 15 RBs 19 UEs merge
id=20 12 RBs 11 UEs merge
id=21 13 RBs 12+1 UEs merge
id=19 15 RBs 19 UEs merge

FIGURE 16. **Slice configuration, Scenario 1, splitting algorithm DSbB, first 7 events.**

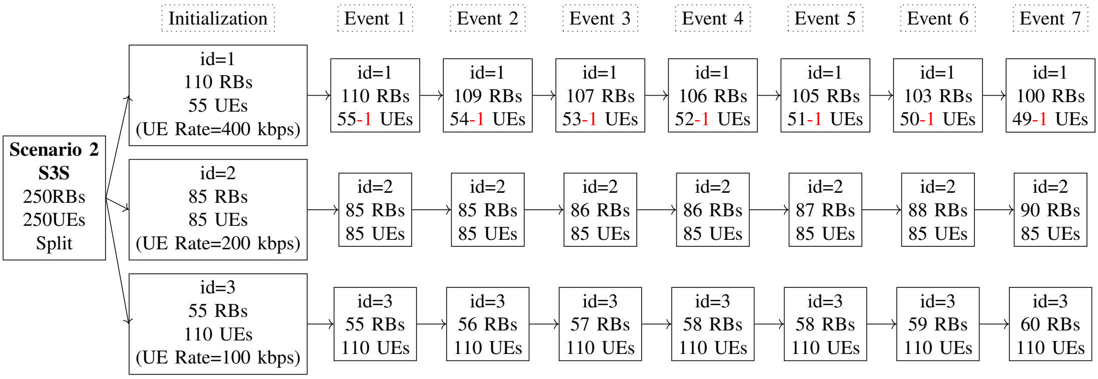


FIGURE 17. **Slice configuration, scenario 2, splitting algorithm S3S, adopted from [5], first 7 events.**

subslice performance dependence of subslice size objective function is not convex or has multiple minimums.

ACKNOWLEDGMENT

During the preparation of this work, the author(s) used Paperpal and LanguageTool in order to improve grammar, spelling and style. After using this tool/service, the author(s)

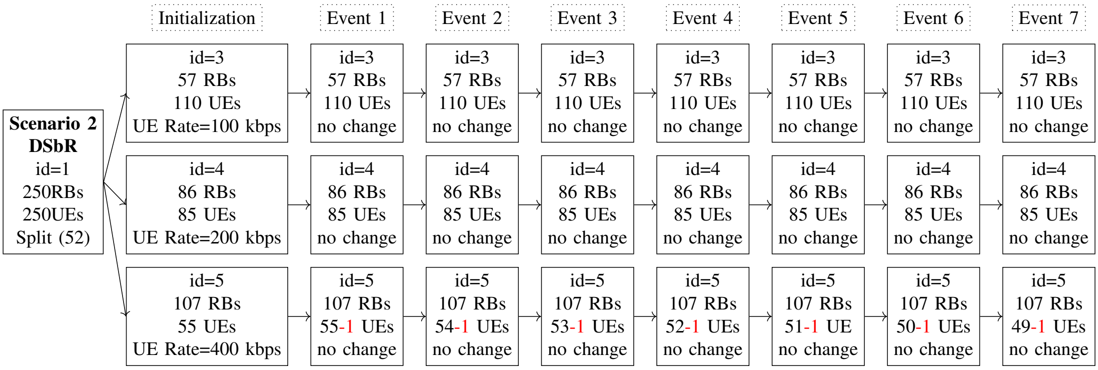

Initialization
Event 1
Event 2
Event 3
Event 4
Event 5
Event 6
Event 7
Scenario 2 DSbR id=1 250RBs 250UEs Split (52)
id=3 57 RBs 110 UEs UE Rate=100 kbps
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=3 57 RBs 110 UEs no change
id=4 86 RBs 85 UEs UE Rate=200 kbps
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=4 86 RBs 85 UEs no change
id=5 107 RBs 55 UEs UE Rate=400 kbps
id=5 107 RBs 55-1 UEs no change
id=5 107 RBs 54-1 UEs no change
id=5 107 RBs 53-1 UEs no change
id=5 107 RBs 52-1 UEs no change
id=5 107 RBs 51-1 UE no change
id=5 107 RBs 50-1 UEs no change
id=5 107 RBs 49-1 UEs no change

FIGURE 18. **Slice configuration, scenario 2, splitting algorithm DSbR, first 7 events.**

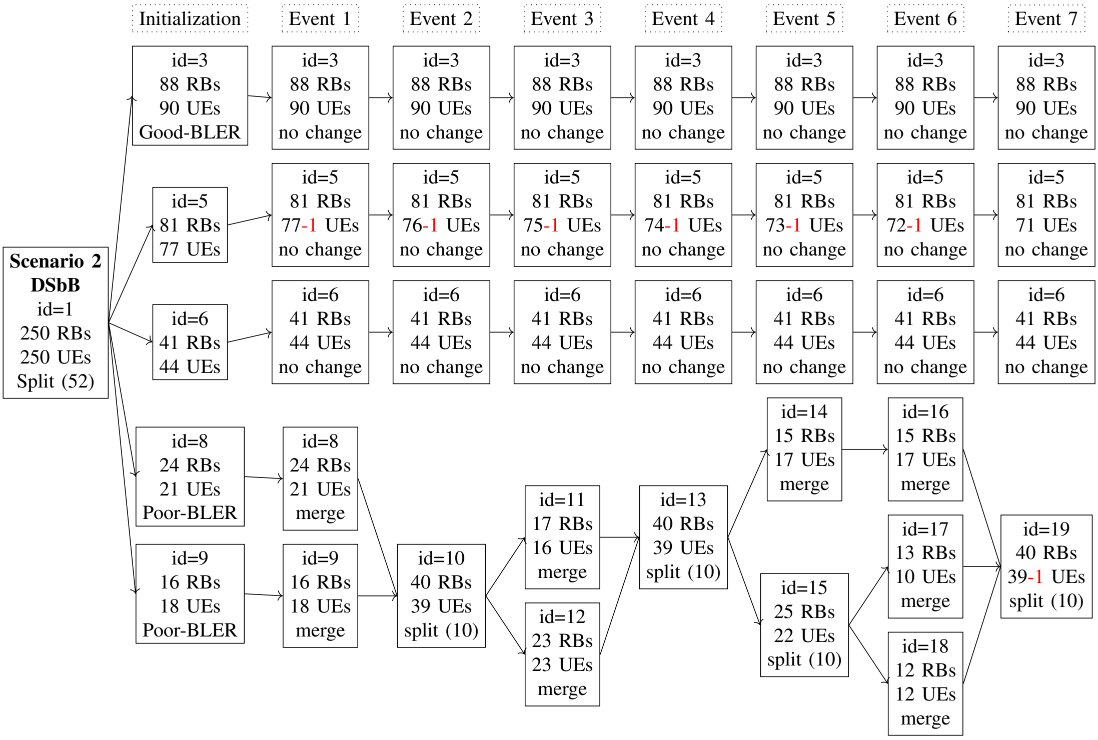

Initialization
Event 1
Event 2
Event 3
Event 4
Event 5
Event 6
Event 7
Scenario 2 DSbB id=1 250 RBs 250 UEs Split (52)
id=3 88 RBs 90 UEs Good-BLER
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=3 88 RBs 90 UEs no change
id=5 81 RBs 77 UEs
id=5 81 RBs 77-1 UEs no change
id=5 81 RBs 76-1 UEs no change
id=5 81 RBs 75-1 UEs no change
id=5 81 RBs 74-1 UEs no change
id=5 81 RBs 73-1 UEs no change
id=5 81 RBs 72-1 UEs no change
id=5 81 RBs 71 UEs no change
id=6 41 RBs 44 UEs
id=6 41 RBs 44 UEs no change
id=6 41 RBs 44 UEs no change
id=6 41 RBs 44 UEs no change
id=6 41 RBs 44 UEs no change
id=6 41 RBs 44 UEs no change
id=6 41 RBs 44 UEs no change
id=6 41 RBs 44 UEs no change
id=8 24 RBs 21 UEs Poor-BLER
id=9 16 RBs 18 UEs Poor-BLER
id=8 24 RBs 21 UEs merge
id=9 16 RBs 18 UEs merge
id=10 40 RBs 39 UEs split (10)
id=11 17 RBs 16 UEs merge
id=12 23 RBs 23 UEs merge
id=13 40 RBs 39 UEs split (10)
id=14 15 RBs 17 UEs merge
id=15 25 RBs 22 UEs split (10)
id=16 15 RBs 17 UEs merge
id=17 13 RBs 10 UEs merge
id=18 12 RBs 12 UEs merge
id=19 40 RBs 39-1 UEs split (10)

FIGURE 19. **Slice configuration, scenario 2, splitting algorithm DSbB, first 7 events.**

reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

REFERENCES

[1] I. Vaishnavi and L. Ciavaglia, "Challenges Towards Automation of Live Telco Network Management: Closed Control Loops," in *2020 16th Int. Conf. Netw. Serv. Manag.*, IEEE, Nov. 2020, pp. 1–5, ISBN: 978-3-903176-31-7, DOI: 10.23919/CNSM50824.2020.9269048.

[2] J. Moysen and L. Giupponi, "From 4G to 5G: Self-organized network management meets machine learn-

ing," *Comput. Commun.*, vol. 129, pp. 248–268, Sep. 2018, ISSN: 01403664, DOI: 10.1016/j.comcom.2018.07.015.

[3] 3GPP, "TS 28.535 - Management and orchestration; Management services for communication service assurance; Requirements," Tech. Rep., 2020, [Online]. Available: https://www.3gpp.org/DynaReport/28535.htm.

[4] 3GPP, "TS 28.530 - Management and orchestration; Concepts, use cases and requirements," Tech. Rep., 2020, [Online]. Available: https://www.3gpp.org/DynaReport/28530.htm.

[5] S. K. Singh, M. M. Salim, J. Cha, Y. Pan, and J. H. Park, "Machine Learning-Based Network Sub-Slicing Framework in a Sustainable 5G Environment," *Sustainability*, vol. 12, no. 15, p. 6250, Aug. 2020, ISSN: 2071-1050, DOI: 10.3390/su12156250.

[6] M. Kulmar, I. Müürsepp, and M. M. Alam, "Heuristic Radio Access Network Subslicing with User Clustering and Bandwidth Subpartitioning," *Sensors*, vol. 23, no. 10, p. 4613, May 2023, ISSN: 1424-8220, DOI: 10.3390/s23104613.

[7] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer (Long. Beach. Calif).*, vol. 36, no. 1, pp. 41–50, Jan. 2003, ISSN: 0018-9162, DOI: 10.1109/MC.2003.1160055.

[8] R. Boutaba, N. Shahriar, M. A. Salahuddin, S. R. Chowdhury, N. Saha, and A. James, "AI-driven Closed-loop Automation in 5G and beyond Mobile Networks," in *Proc. 4th FlexNets Work. Flex. Networks Artif. Intell. Support. Netw. Flex. Agil.*, New York, NY, USA: ACM, Aug. 2021, pp. 1–6, ISBN: 9781450386340, DOI: 10.1145/3472735.3474458.

[9] N. F. Saraiva de Sousa and C. E. Rothenberg, "CLARA: Closed Loop-based Zero-touch Network Management Framework," in *2021 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Networks*, IEEE, Nov. 2021, pp. 110–115, ISBN: 978-1-6654-3983-1, DOI: 10.1109/NFV-SDN53031.2021.9665048.

[10] M. Xie, P. H. Gomes, J. Harmatos, and J. Ordonez-Lucena, "Collaborated Closed Loops for Autonomous End-to-End Service Management in 5G," in *2020 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Networks*, IEEE, Nov. 2020, pp. 64–70, ISBN: 978-1-7281-8159-2, DOI: 10.1109/NFV-SDN50289.2020.9289902.

[11] H. Chergui, A. Ksentini, L. Blanco, and C. Verikoukis, "Toward Zero-Touch Management and Orchestration of Massive Deployment of Network Slices in 6G," *IEEE Wirel. Commun.*, vol. 29, no. 1, pp. 86–93, Feb. 2022, ISSN: 1536-1284, DOI: 10.1109/MWC.009.00366.

[12] M. Gramaglia, M. Kajo, C. Mannweiler, Ö. Bulakci, and Q. Wei, "A unified service-based capability exposure framework for closed-loop network automation," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 11, Nov. 2022, ISSN: 2161-3915, DOI: 10.1002/ett.4598.

[13] M. Mekki, S. Arora, and A. Ksentini, "A Scalable Monitoring Framework for Network Slicing in 5G and Beyond Mobile Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 1, pp. 413–423, Mar. 2022, ISSN: 1932-4537, DOI: 10.1109/TNSM.2021.3119433.

[14] S. Kuklinski and L. Tomaszewski, "Key Performance Indicators for 5G network slicing," in *2019 IEEE Conf. Netw. Softwarization*, IEEE, Jun. 2019, pp. 464–471, ISBN: 978-1-5386-9376-6, DOI: 10.1109/NETSOFT.2019.8806692.

[15] E. Pateromichelakis, F. Moggio, C. Mannweiler, *et al.*, "End-to-End Data Analytics Framework for 5G Architecture," *IEEE Access*, vol. 7, pp. 40 295–40 312, 2019, ISSN: 2169-3536, DOI: 10.1109/ACCESS.2019.2902984.

[16] 3GPP, "TS 38.104 - NR; Base Station (BS) radio transmission and reception," 2022, [Online]. Available: https://www.3gpp.org/DynaReport/38104.htm.

[17] MathWorks, *NR Cell Performance Evaluation with Physical Layer Integration*, 2022, [Online]. Available: https://se.mathworks.com/help/5g/ug/nr-cell-performance-evaluation-with-physical-layer-integration.html (visited on 01/19/2022).

[18] 3GPP, "TS 38.901- Radio Access Network; Study on channel model for frequencies from 0.5 to 100 GHz," Tech. Rep., 2019, [Online]. Available: https://www.3gpp.org/DynaReport/38901.htm.

[19] 3GPP, "TS 38.211 - NR; Physical channels and modulation," 2020, [Online]. Available: https://www.3gpp.org/DynaReport/38211.htm.

[20] M. Kulmar, I. Müürsepp, and M. M. Alam, "Enhanced Decision Mechanism for RAN Subslicing in Management Closed Control Loop," in *2023 Eighth Int. Conf. Fog Mob. Edge Comput.*, IEEE, Sep. 2023, pp. 175–181, ISBN: 979-8-3503-1697-1, DOI: 10.1109/FMEC59375.2023.10306223.

Curriculum Vitae

1. Personal data

Name	Marika Kulmar
Nationality	Estonian

2. Contact information

Address	Tallinn University of Technology, School of Information Technology, Thomas Johann Seebeck Department of Electronics, Ehitajate tee 5, 19086 Tallinn, Estonia
E-mail	Marika.Kulmar@taltech.ee

3. Education

2020–…	Tallinn University of Technology, School of Information Technologies, Information and Communication Technology, PhD studies
1999–2002	Tallinn University of Technology, School of Information Technologies, Department of Radio and Communication Engineering, Telecommunication, MSc
1994–1999	Tallinn University of Technology, School of Information Technologies, System engineering, diploma

4. Language competence

Estonian	native
English	fluent

5. Professional employment

2020– …	Tallinn University of Technology, School of Information Technologies, Thomas Johann Seebeck Department of Electronics, Junior Researcher
2017–2020	Tallinn University of Technology, School of Information Technologies, Thomas Johann Seebeck Department of Electronics, Lecturer
2007–2016	Tallinn University of Technology, School of Information Technologies, Department of Radio and Communication Engineering, Teaching assistant
2000–2006	Tallinn University of Technology, School of Information Technologies, Department of Radio and Communication Engineering, Engineer
1996–2000	Tallinn University of Technology, Computer Centre, Engineer

6. Computer skills

- Operating systems: Windows, Linux, MacOS
- Document preparation: Microsoft Office, LibreOffice, Latex
- Scientific packages: MATLAB

Elulookirjeldus

1. Isikuandmed

Nimi	Marika Kulmar
Kodakondsus	Eesti

2. Kontaktandmed

Aadress	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond Thomas Johann Seebecki elektroonikainstituut, Ehitajate tee 5, 19086 Tallinn, Estonia
E-post	Marika.Kulmar@taltech.ee

3. Haridus

2020–. . .	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Info- ja kommunikatsioonitehnoloogia, doktoriõpe
1999–2002	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Raadio- ja sidetehnika instituut, Telekommunikatsioon, MSc
1994–1999	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Süsteemitehnika, diplom

4. Keelteoskus

eesti keel	emakeel
inglise keel	kõrgtase

5. Teenistuskäik

2020– . . .	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Thomas Johann Seebecki elektroonikainstituut, doktorant-nooremteadur
2017–2020	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Thomas Johann Seebecki elektroonikainstituut, lektor
2007–2016	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Raadio- ja sidetehnika instituut, assistent
2000–2006	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Raadio- ja sidetehnika instituut, insener
1996–2000	Tallinna Tehnikaülikool, Arvutuskeskus, insener

6. Arvutioskused

- Operatsioonisüsteemid: Windows, Linux, MacOS
- Kontoritarkvara: Microsoft Office, LibreOffice, Latex
- Teadustarkvara paketid: MATLAB

ISSN 2585-6901 (PDF)
ISBN 978-9916-80-097-3 (PDF)