11-9-2023 10:45 AM

Towards Zero Touch Next Generation Network Management

sam aleyadeh, *Western University*

Supervisor: Shami, Abdallah, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Electrical and Computer Engineering

11-9-2023 10:45 AM

Towards Zero Touch Next Generation Network Management

sam aleyadeh

Supervisor: Shami, Abdallah, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Electrical and Computer Engineering

Abstract

The current trend in user services places an ever-growing demand for higher data rates, near-real-time latencies, and near-perfect quality of service. To meet such demands, fundamental changes were made to the front and mid-haul and backbone networking segments servicing them. One of the main changes made was virtualizing the networking components to allow for faster deployment and reconfiguration when needed. However, adopting such technologies poses several challenges, such as improving the performance and efficiency of these systems by properly orchestrating the services to the ideal edge device. A second challenge is ensuring the backbone optical networking maximizes and maintains the throughput levels under more dynamically variant conditions. A third challenge is addressing the limitation of placement techniques in O-RAN. In this thesis, we propose using various optimization modeling and machine learning techniques in three segments of network systems towards lowering the need for human intervention targeting zero-touch networking. In particular, the first part of the thesis applies optimization modeling, heuristics, and segmentation to improve the locally driven orchestration techniques, which are used to place demands on edge devices throughput to ensure efficient and resilient placement decisions. The second part of the thesis proposes using reinforcement learning (RL) techniques on a nodal base to address the dynamic nature of demands within an optical networking paradigm. The RL techniques ensure blocking rates are kept to a minimum by tailoring the agents' behavior based on each node's demand intake throughout the day. The third part of the thesis proposes using transfer learning augmented reinforcement learning to drive a network slicing-based solution in O-RAN to address the stringent and divergent demands of 5G applications. The main

contributions of the thesis consist of three broad parts. The first is developing optimal and heuristic orchestration algorithms that improve demands' performance and reliability in an edge computing environment. The second is using reinforcement learning to determine the appropriate spectral placement for demands within isolated optical paths, ensuring lower fragmentation and better throughput utilization. The third is developing a heuristic controlled transfer learning augmented reinforcement learning network slicing in an O-RAN environment. Hence ensuring improved reliability while maintaining lower complexity than traditional placement techniques.

Keywords: Edge Computing, Orchestration, Segmentation, Mobility, Clustering, 5G, Elastic Optical Networks, Spectrum Fragmentation, Deep Reinforcement Learning, Transfer Learning, O-RAN, Network Slicing.

Summary for Lay Audience

As the use of smart and connected devices continues to grow, there is a need for more reliable and faster networking infrastructure. This thesis focuses on addressing the challenges faced by 5G systems and their underlying infrastructure in three domains: user-adjacent edge computing, Open Radio Access Networks (O-RANs), and back-end Optical networking. The main goal is to automate these networks to improve their reliability and speed. To achieve this, the thesis presents optimization algorithms, heuristic strategies, and machine-learning models to enhance the performance of network orchestration, slicing, and spectrum allocation. By implementing these contributions, network service providers can offer more reliable, secure, and fast services. The resulting networks can support more diverse applications with increasing demands such as Autonomous Vehicles, Augmented Reality, Industrial Automation, Emergency Services, and Smart Grids. This thesis paves the way for a new era of networking with automated organization and self-heal capabilities that target zero-touch networks, where no human interaction is needed to maintain or fix the networks in cases of failures.

Co-Authorship

The following thesis contains material from previously published papers and manuscripts submitted for publication that have been co-authored by Dr. Abdallah Moubayed, Prof. Abdallah Shami, Dr. Parisa Heidari, Dr. Abbas Javadtalab, and Ibrahim Tamim. All the research, developments, simulations, and work presented here were carried out by Sam Aleyadeh under the guidance of Dr. Abdallah Shami. Chapters 3-7 are based on the following manuscripts that have been submitted, accepted, or published.

1. Sam Aleyadeh, Abdallah Moubayed, Parisa Heidari, and Abdallah Shami, "Optimal container migration/re-instantiation in hybrid computing environments" in the IEEE Open Journal of the Communications Society (OJCOM), 2022.

2. Sam Aleyadeh, Abdallah Moubayed, and Abdallah Shami, "Mobility aware edge computing segmentation towards localized orchestration," in the International Symposium on Networks, Computers and Communications (ISNCC), 2021.

3. Sam Aleyadeh, Abbas Javadtalab, and Abdallah Shami, "Modular Simulation Environment Towards OTN AI-based Solutions," in the International Conference on Intelligent Data Science Technologies and Applications (IDSTA), 2023.

4. Sam Aleyadeh, Abbas Javadtalab, and Abdallah Shami "Throughput Latency Targeted RL Spectrum Allocation In Heterogeneous OTN," in the Annals of Telecommunications, 2023. (In Submission).

5. Sam Aleyadeh, Ibrahim Tamim, and Abdallah Shami "Transfer Learning Accelerated Network Slice Management for Next Generation Services," in the International Journal for the Computer and Telecommunications Industry, 2023. (In Submission).

Acknowledgements

First and foremost, I give my thanks to Allah; nothing would have been achievable without his guidance and blessing, to whom all praise is due. I am deeply thankful to Allah for giving me the fortitude and resilience needed to accomplish this thesis.

Second, I am profoundly grateful to my supervisor, Dr. Abdallah Shami, whose unwavering support, motivation, and mentorship throughout my doctoral journey were invaluable. His commitment and guidance were not only inspirational but also instrumental in the successful completion of this thesis. Without his continuous guidance, this work could not have been accomplished.

Third, I would like to thank my examination committee, namely Dr. Anwar Haque, Dr. Xianbin Wang, Dr. Ahmed Refaey Hussein, and Dr. Salimur Choudhury. Thank you for taking the time to review and examine my thesis. I would also like to thank my Master's thesis supervisor Dr. Hossam Hussanain for carefully guiding me on the first steps of my research career. Also, a huge thank you to Dr. Abd-Elhamid M. Taha and Dr. Abdallah Moubayed for the support they have given me throughout my Ph.D. journey.

Additionally, I would like to thank Courtney Harper and all the administrative staff at Western University for providing me with all the help throughout this journey.

And last but certainly not least, My deepest love and gratitude are dedicated to my mother, Dr. Najah Abu Ali. Her relentless support, especially in times of self-doubt, enduring patience, and infinite affection, have been my life's bedrock. Life without her is simply unimaginable; to my siblings, who constantly encouraged me to transcend my perceived boundaries and have always believed in my potential. And to my wife, Lujain Haddad, the love of my life, for her unwavering support throughout this journey. Her unconditional love, support in realizing my dreams, and inspiration for my personal growth are deeply cherished.

Sam Aleyadeh

To the memory of Jameela bader

Table of Contents

List of Tables

Section **Page**

List of Figures

List of Abbreviations

5G	5th Generation Mobile Network
CAPEX	CAPital EXpenditure
OPEX	OPerational EXpenditure
VMs	Virtual Machines
RSUs	RoadSide Units
QoS	Quality of Service
NFV	Network Function Virtualization
IP	Integer Programming
VNFs	Virtual Network Functions
IP	Integer Programming
ECC	Edge Cloud Computing
EC	Edge Computing
SDNs	Software-Defined Networks
NSPs	Networks Service Providers
AI	Artificial Intelligence
OTNs	Optical Transport Networks
NR	New Radio
eMBB	enhanced Mobile BroadBand
URLLC	Ultra-Reliable Low Latency Communication
mMTC	massive Machine-Type Communications
WDM	Wavelength Division Multiplexing
RWA	Routing and Wavelength Assignment
RSA	Routing and Spectrum Assignment
SA	Spectrum Allocation
VR	Virtual Reality
AR	Augmented Reality
NGNs	Next-Generation Networks
RL	Reinforcement Learning
ML	Machine Learning
BBR	Bandwidth Blocking Ratio

RAN	Radio Access Network
CRAN	Centralized RAN
O-RAN	Open RAN
near-RT RIC	near-Real-Time RAN Intelligent Controller
TS	Traffic Steering
SP	Service Placement
TL	Transfer Learning
SFC	Service Function Chain
MINLP	Mixed-Integer NonLinear Programming
MADRL	Multi-Agent Deep Reinforcement Learning
ZSM	Zero-touch network and Service Management
AVL	Automatic Vehicle Locator
V2X	Vehicle-to-Everything
B5G	Beyond 5G
IoT	Internet of Things
O-CU	O-RAN Centralized Unit
O-DU	O-RAN Distributed Unit
COTS	Commercial-Off-the-Shelf servers
MTTF	Mean Time To Failure

Chapter 1
Introduction

The 5th Generation Mobile Network (5G) offers enhanced connectivity, higher data transfer speeds, lower latency, and increased capacity compared to its predecessors. The adoption of 5G technology has been accelerated, leading to a significant increase in the number of 5G devices in recent years. This can be seen in the illustrated Figure 1.1[2]. Additionally, thanks to the more reliable connections offered by 5G Enhanced Mobile Broadband (eMBB), we have seen a significant increase in the number of 5G applications available, making it an indispensable aspect of our everyday routines. Hence, users can enjoy a better mobile browsing experience, video streaming, and immersive virtual and augmented reality involvement. Additionally, 5G's impact on industrial applications is significant due to its massive Machine-Type Communications (mMTC), which enables more devices to connect to the network, making it ideal for monitoring and maintaining factory equipment. Furthermore, 5G's Ultra-Reliable and Low Latency Communications (URLLC) capabilities allow for critical real-time applications such as autonomous vehicles and remote surgery, where even the slightest delay could have severe consequences. Consequently, with the increased adoption of 5G-based services, the amount of traffic and demands will inevitably increase. This will be especially true with the growing deployment of Smart Cities and their infrastructure, as well as Public Safety and Surveillance.

Human control is insufficient to maintain and accommodate the above applications. Hence, automation in 5G, also called Zero-touch becomes more crucial. This can be attributed to several reasons, such as the efficiency and speed the zero-touch provides with the increased complexity and scale of networks using technologies such as network slicing. Zero-touch also offers several additional advantages, such as Reduced Operational Expenses (OPEX) since the automated systems lower the need for human intervention. This greatly decreases the costs associated with network management and makes the services more cost-effective. Improving reliability and consistency is important to reduce

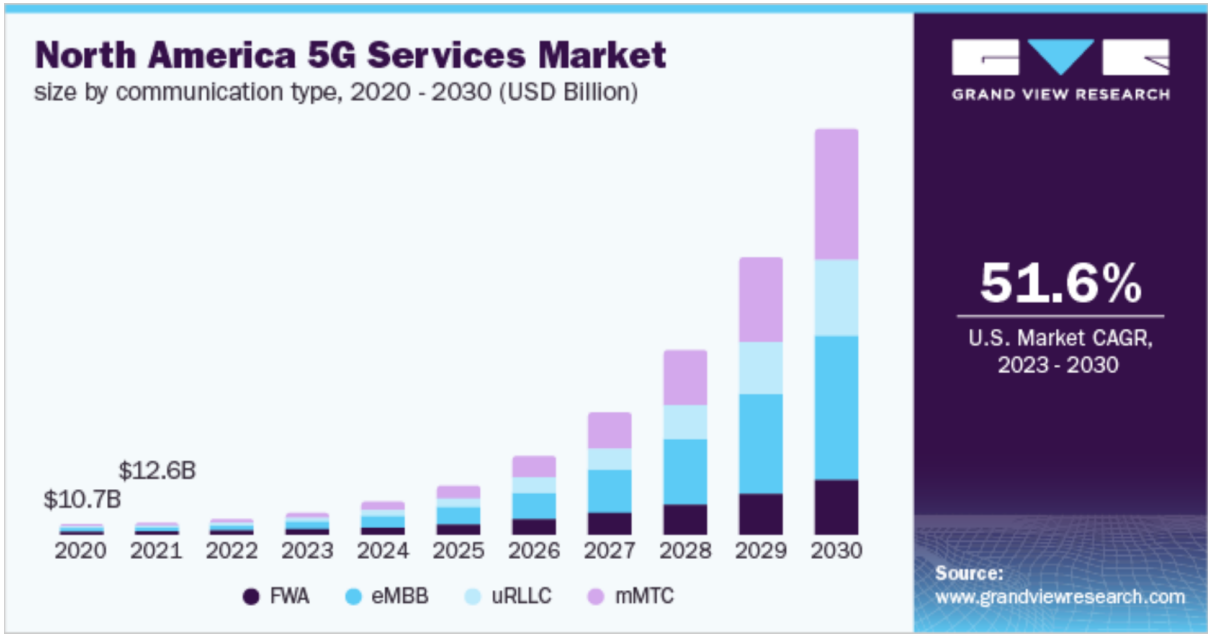

Figure 1.1: 5G Growth Trend

the chances of human error, which is a common cause of network outages. Finally, zero-touch allows for Real-Time Management and Optimization. With zero-touch automation, networks can be managed and optimized in real-time. This is especially important in 5G, where network conditions can change rapidly, for instance, due to the mobility of users or rapid fluctuations in traffic demand.

1.1 Motivation

The networking infrastructure is facing a significant surge in complexity due to the rapid growth of the 5G platform. This complexity arises from various factors, including expanding network scale driven by the proliferation of connected devices, the devices dispersed geographical locations, the high diversity of the devices' computational capabilities, and heterogeneity of the network technologies, protocols, and infrastructure with each network paradigm having its own specifications and requirements. For instance, many organizations have adopted edge, hybrid, and multi-cloud architectures that combine private and public cloud services, exemplifying the diversity present. Furthermore, deploying Software-Defined Networking (SDN) for decentralized network management and control has introduced additional layers of abstraction and complexity when con-

figuring and orchestrating network services, aiming to achieve enhanced flexibility and programmability.

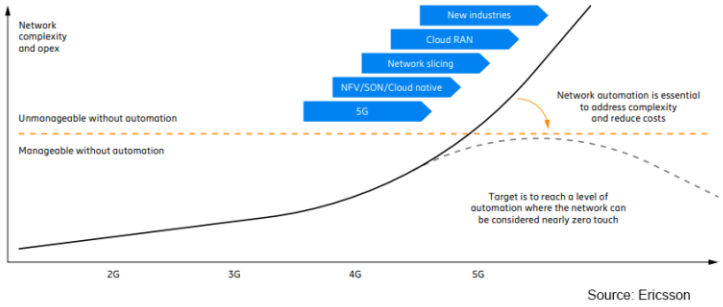

Figure 1.2: 5G Complexity

In order to take full advantage of the benefits of 5G networks, it is crucial to reduce the complexity of network management. That's why effective automation techniques are seen as a highly appealing solution. These techniques address the challenges arising from the lack of human supervision, network management overhead delays, and associated capital and operational expenses (CAPEX and OPEX). The industry and academia are in a continuous quest to enable automation in network infrastructure, leading to the emergence of the concept of zero-touch networking. However, The automation process has become more challenging because of the different solutions geared towards specific fields, which goes against its intended goal, as depicted in Figure 1.2 [3].

Automation has become an essential tool for handling the expanding scale of networking infrastructure. This includes various domains such as edge computing, O-RAN, and OTN, each with its own set of challenges and goals. As growth trends continue, automation will be increasingly important in managing these domains.

Automation has become an essential tool in recent years for handling the expanding scale of networking infrastructure. This includes various network infrastructure domains and scales, ranging from user-adjacent front haul (e.g., edge computing) to mid-haul (e.g., O-RAN) and backbone (e.g., OTN). Each with its own set of challenges and goals. As growth trends continue, automation will be increasingly important in managing these domains.

1.2 Thesis Objectives

This thesis aims to propose automation schemes grouped into three projects. Each project aims to enable zero-touch management in a specific network infrastructure domain. The first project proposes automating orchestration mechanisms for edge computing applications. The second project steered toward simulating and optimizing Optical networks spectrally. The third project focuses on managing network slices in the O-RAN infrastructure utilizing AI techniques and using real-life captured 5G traffic datasets.

The first project presents an optimization scheme and a heuristic scheme to effectively manage edge computing containers by making efficient decisions on whether to instantiate or migrate them. An intelligent segmentation scheme is proposed to better split the edge computing solution space, allowing for effective local management and orchestration. This scheme addresses communication overhead issues that are typical in centralized management platforms.

The second project aims to develop a comprehensive and highly modular simulation setup, which helps generate datasets of 5G demands in an optical environment, which is a scarce resource. The generated dataset is used to train a reinforcement learning model that can organize the demands within the spectral space at each node. The objective is to improve throughput while respecting latency requirements.

The third project aims to incorporate 5G into O-RAN. It also strives for addressing the optimal placement of Virtual Network Functions (VNFs) to efficiently support the three main types of 5G applications. The project utilizes machine learning techniques to create intelligent agents that are capable of designing network slices while prioritizing minimized downtime over extended periods of time. The approach aims to simplify the placement process by employing network slices instead of managing 5G applications individually.

1.3 Thesis Organization

This thesis presents various techniques and tools for automating and improving 5G services and beyond. Our focus is on the front haul, the mid-haul, and the backbone network infrastructure, with the ultimate goal of achieving a zero-touch framework. We

present our findings in an integrated article format. We proceed by presenting the background of the tools and techniques used to develop our various solutions in Chapter 2. Chapter 3 presents an orchestration solution using optimization and heuristic techniques. Chapter 4 presents a novel edge computing segmentation scheme geared towards creating independent sectors capable of self-orchestration and management. Chapter 5 presents a modular OTN-enabled 5G end-to-end simulator capable of operating in a distributed environment. Chapter 6 presents a scheme that utilizes Reinforcement Learning to optimize throughput by creating agents on every node in an optical network. The scheme sole purpose is to reduce fragmentation and prevent demand blocking. In Chapter 7, we investigate the challenge that 5G faces in O-RAN regarding VNF assignment. We introduce an intelligent network slice management system that uses a Transfer Learning augmented Deep Reinforcement Learning model to address this issue. Chapter 8 concludes the thesis and provides some insights into future research directions.

1.4 Thesis Contributions

The major contributions of the thesis are summarized as follows.

1.4.1 Contributions of Chapter 3

1. We designed a scheme for service orchestration in an edge computing environment to reduce the containers' downtime and latency. The problem is formulated as a mixed integer linear programming problem (MILP). However, the complexity of the MILP increases with the increase in the number of containers. Thus to tackle this issue,

2. we subdivided the solution space into smaller regions to allow for a suboptimal practical heuristic solution with reduced complexity and,

3. designed a suboptimal heuristic algorithm to enable automation in zero-touch.

1.4.2 Contributions of Chapter 4

1. To minimize communication overhead in the aforementioned orchestration schemes, we have devised a segmentation scheme to facilitate local orchestration. Hence, achieving a lower complexity and improving network scalability.

2. To account for mobility and network heterogeneity, we designed a Virtual localization technique to improve the segmentation technique. Hence, improving stability and accuracy.

3. We utilized Lax Clustering techniques to create unique but not necessarily geometrically shaped clusters instead of grid-like approaches. Hence, delivering practical solutions for implementation in real-life networks.

1.4.3 Contributions of Chapter 5

1. Introduced an end-to-end simulation environment to generate datasets. Hence, addressing the lack of OTN 5G datasets.

2. Developed a modular end-to-end simulator that interfaces well-known standard simulators such as NS3, SOMO, OMNINET, and NetSim. Hence, allow for better utilization of the capabilities of these simulators based on researcher needs.

3. The end-to-end simulator enables distributed operation of the simulation models to be operated across multiple machines. Hence, reduce the hardware requirements by dispersing the workload across several machines, rather than relying on a single powerful workstation. This enables the simulation to run more efficiently.

1.4.4 Contributions of Chapter 6

1. Formulated the problem of spectral allocation in an optical networking environment as a reinforcement learning model.

2. Implemented the anchoring of demands, forced allocation direction, and traditional networks to further optimize the RL model.

1.4.5 Contributions of Chapter 7

1. Addressed the service placement problem in an O-RAN environment via an intelligent network-slicing solution.

2. Presented a Transfer Learning augmented Deep Reinforcement Learning model to create the 5G tailored network slices.

3. To automate the Reinforcement Learning model, we formulated a heuristic driver enabling zero-touch management within O-RAN

Chapter 2
Background

2.1 Introduction

As mentioned earlier, the rapid introduction of new technology replacing archaic networking infrastructure has created highly complex interactions that resulted in the need for efficient algorithms to help maintain the performance benefits brought about by those systems and processes. Moreover, these systems need to be able to make intelligent decisions to further enhance their performance and stability. To that end, this thesis proposes the use of linear programming, clustering, and machine learning techniques to improve the performance of different considered systems and processes. In what follows, a brief background about the different types of linear programming problems and machine learning techniques is given. Section 2.2 presents the different types of linear models including standard, Integer Linear Programming, Mixed-Integer Linear Programming, and other models. Furthermore, Section 2.3 discusses the different types of machine learning algorithms including supervised, unsupervised, transfer, reinforcement, and deep learning. Moreover, Section 2.4 also discusses the concept of networking slicing and its impact on emerging networking paradigms.

2.2 Linear Programming

Linear programming is a powerful mathematical technique used to optimize resource allocation and decision-making processes in various fields, including operations research, economics, engineering, and management. It provides a systematic approach to solve complex problems involving limited resources and multiple objectives by formulating them into a mathematical model.

At its core, linear programming deals with maximizing or minimizing a linear objective function subject to a set of linear constraints. The objective function represents

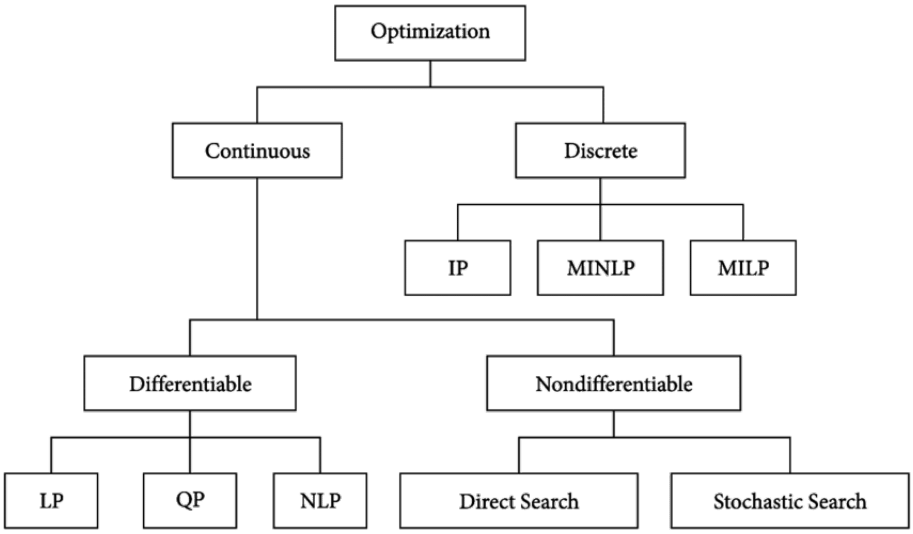

Figure 2.1: Linear Programming types

the quantity to be optimized, such as profit, cost, time, or efficiency, while the constraints define the limitations or restrictions imposed on the decision variables.

There are three main types of linear programming problems, each characterized by the type of constraints and variables involved. These types are:

1. Standard Linear Programming: This is the most basic type of linear programming problem, also known as Continuous Linear Programming. In this type, all decision variables are continuous and can take any real value within a specified range or domain. The constraints are represented by a system of linear inequalities or equations. The objective is to optimize a linear objective function subject to these constraints. The simplex method and other linear programming algorithms are commonly used to solve standard linear programming problems.

2. Integer Linear Programming (ILP): Integer Linear Programming extends the standard linear programming by adding the requirement that all decision variables must be integers. This type of problem is useful when the decision variables represent quantities that must be discrete or combinatorial in nature, such as the number of units produced, the selection of projects or routes, or the assignment of workers to shifts. ILP problems are generally more challenging to solve than standard linear programming due to the combinatorial nature of integer variables. Specialized algorithms, such as the branch-and-bound method, are commonly employed to find optimal solutions.

3. Mixed-Integer Linear Programming (MILP): Mixed-Integer Linear Programming

combines the features of standard linear programming and integer linear programming. In MILP, some decision variables are allowed to be continuous, while others are required to be integers. This type of problem is suitable when a combination of continuous and discrete decisions needs to be made. The inclusion of integer variables adds complexity to the problem, and efficient algorithms, such as branch-and-bound, are used to navigate the solution space and find optimal solutions.

In addition to these main types, there are variations and special cases of linear programming, such as binary linear programming (where decision variables are restricted to take only binary values of 0 or 1), multi-objective linear programming (where multiple objective functions are simultaneously optimized), and stochastic linear programming (where uncertainty or randomness is considered in the constraints or objective function). These variations cater to specific problem characteristics and requirements, providing further flexibility in modeling and solving optimization problems.

2.2.1 Integer Linear Programming

Integer Linear Programming (ILP) is a branch of mathematical optimization that deals with optimization problems where all of the decision variables are required to be integers. It extends the concepts of linear programming to include the additional constraint of integrality, making it more suitable for problems that involve discrete or combinatorial decision-making.

In Integer Linear Programming, the objective is still to optimize a linear objective function, subject to a set of linear constraints. The objective function represents the quantity to be maximized or minimized, such as profit, cost, or time. The decision variables in ILP can take on discrete values, usually integers, rather than continuous values. The constraints define the limitations or restrictions on the decision variables, which can include both linear equations and inequalities.

The inclusion of integer variables in ILP problems adds a layer of complexity compared to linear programming. The presence of discrete values in the decision variables leads to a more challenging problem of finding feasible and optimal solutions. The search space for ILP problems is often much larger than that of linear programming due to the combinatorial nature of integer variables.

Solving ILP problems requires specialized algorithms that can efficiently explore the solution space to identify the optimal integer solutions. The branch-and-bound method, combined with various heuristics, is commonly used to solve ILP problems. It divides the solution space into smaller subspaces, evaluates the objective function within each branch, and progressively prunes branches that cannot contain better solutions than the current best solution found.

ILP problems find applications in numerous areas, such as production planning, scheduling, logistics, network design, and resource allocation. They are particularly useful in problems that involve discrete decisions, such as assigning workers to shifts, selecting project alternatives, determining the optimal routing of vehicles, or scheduling tasks with precedence constraints.

ILP has proven to be a valuable tool for decision-makers in both industry and academia. By incorporating the requirement for integer solutions, ILP provides more accurate and realistic models for many real-world problems. It enables decision-makers to make optimal choices while considering the discrete nature of the decision variables, leading to improved resource allocation, cost reduction, and overall efficiency in various domains.

2.2.2 Mixed-Integer Linear Programming

Mixed-Integer Linear Programming (MILP) is an extension of linear programming that allows for the inclusion of integer variables in addition to continuous variables. It is a powerful mathematical optimization technique used to solve problems where some decision variables must take on discrete or binary values, while others remain continuous. MILP finds applications in various fields, including operations research, logistics, scheduling, finance, and engineering.

In MILP, the objective is still to optimize a linear objective function, but with the added complexity of integer or binary variables. The objective function represents the quantity to be maximized or minimized, such as profit, cost, or time, and is subject to a set of linear constraints. These constraints define the limitations or restrictions imposed on the decision variables, which can now be a mix of continuous and discrete values.

The inclusion of integer variables in MILP problems introduces combinatorial complexity, as the search space expands exponentially. Solving MILP problems requires

exploring a large number of potential solutions to identify the optimal one. The challenge lies in efficiently navigating this vast solution space to find the best feasible solution that satisfies all the constraints and optimizes the objective function.

The branch-and-bound method is a common technique used to solve MILP problems. It systematically divides the solution space into smaller subspaces, called branches, and evaluates the objective function within each branch. By progressively pruning branches that cannot contain a better solution than the current best solution found so far, the algorithm narrows down the search space until it finds the optimal solution.

MILP problems can model a wide range of real-world scenarios. For example, in production planning, MILP can be used to optimize workforce scheduling, equipment allocation, and production sequencing, taking into account factors such as capacity constraints, production costs, and delivery deadlines. In transportation and logistics, it can help optimize route planning, vehicle assignment, and inventory management, considering variables like transportation costs, customer demands, and vehicle capacity.

MILP is also commonly used in financial planning and portfolio optimization. It can aid in asset allocation decisions, investment selection, and risk management by considering discrete decisions, such as the inclusion or exclusion of specific assets or investment options.

2.3 Machine Learning

Machine learning, a sub-field of artificial intelligence, focuses on developing algorithms and models that allow computer systems to learn and make predictions or decisions without explicit programming. It involves the use of statistical techniques and mathematical models to enable computers to automatically learn from data and improve their performance over time. Machine learning algorithms analyze and identify patterns, trends, and relationships in data to generate insights or make predictions. By leveraging large datasets and computational power, machine learning enables computers to learn from experience, adapt to changing conditions, and make accurate predictions or decisions in various domains, ranging from image recognition and natural language processing to recommendation systems and autonomous vehicles.

There are several types of machine learning algorithms, each with its own approach and characteristics. The main types of machine learning algorithms are:

1. Supervised Learning: In supervised learning, the algorithm is trained on a labeled dataset, where the input data is accompanied by corresponding output labels or target values. The goal is to learn a mapping function that can predict the output labels for new, unseen input data. Common algorithms in supervised learning include decision trees, random forests, support vector machines (SVM), naive Bayes, and neural networks.

2. Unsupervised Learning: Unsupervised learning algorithms are used when the input data is unlabeled or lacks explicit output labels. The goal is to identify patterns, structures, or relationships within the data. Common unsupervised learning algorithms include clustering algorithms like k-means, hierarchical clustering, and DBSCAN, as well as dimensionality reduction techniques such as principal component analysis (PCA) and t-SNE.

3. Semi-Supervised Learning: Semi-supervised learning combines elements of supervised and unsupervised learning. It is used when only a subset of the input data is labeled. The algorithm leverages both labeled and unlabeled data to learn patterns and make predictions. Semi-supervised learning can be useful when labeling data is expensive or time-consuming.

4. Reinforcement Learning: Reinforcement learning involves an agent learning to interact with an environment and maximize its cumulative reward. The agent takes actions in the environment, receives feedback in the form of rewards or penalties, and adjusts its behavior based on the feedback to achieve a specific goal. Algorithms such as Q-learning and deep Q-networks (DQN) are commonly used in reinforcement learning.

5. Deep Learning: Deep learning is a subset of machine learning that focuses on neural networks with multiple layers (deep neural networks). These networks can automatically learn hierarchical representations of data and extract complex features. Deep learning has achieved significant breakthroughs in areas such as image recognition, natural language processing, and speech recognition. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are commonly used in deep learning.

6. Transfer Learning: Transfer learning involves leveraging knowledge or models learned from one task or domain to improve performance on another related task or

domain. It allows models to generalize from previous experiences and adapt to new scenarios with less data or training time.

7. Ensemble Learning: Ensemble learning combines multiple models or algorithms to make predictions or decisions. By aggregating the predictions or outputs of individual models, ensemble methods can improve accuracy and robustness. Examples of ensemble learning methods include bagging (e.g., random forests) and boosting (e.g., AdaBoost, Gradient Boosting).

These are some of the main types of machine learning algorithms, each suited for different types of tasks, data characteristics, and learning objectives. Choosing the appropriate algorithm depends on the specific problem at hand and the available data.

2.3.1 Clustering

Clustering is a fundamental technique in data analysis and machine learning that involves grouping similar data points together based on their inherent characteristics or patterns. It aims to identify similarities and differences within a dataset and organize it into meaningful groups, called clusters. Clustering provides valuable insights into the underlying structure of data, facilitates data exploration, and supports decision-making processes.

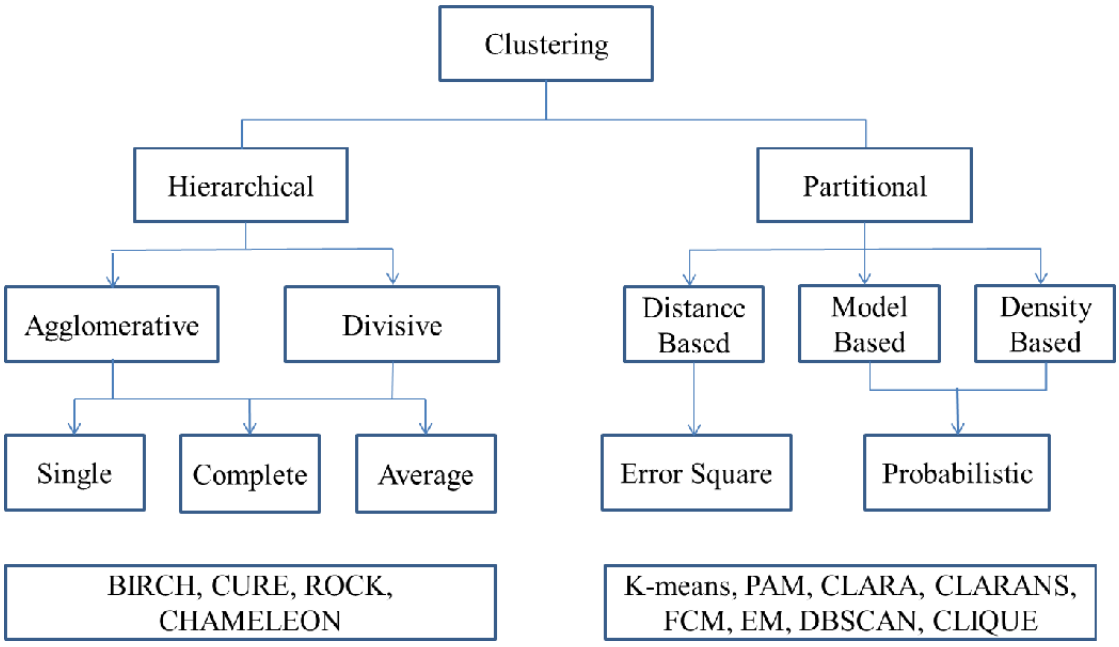

Figure 2.2: Clustering techniques types

There are various clustering algorithms, each with its own approach and assumptions about the data. In this introduction, we will focus on three commonly used clustering techniques: Lax and Strict Radial Clustering and K-means Clustering.

Lax and Strict Radial Clustering: Lax and Strict Radial Clustering are density-based clustering algorithms that identify clusters based on the density of data points in the feature space. These algorithms assume that clusters are formed around high-density regions separated by low-density regions.

In Lax Radial Clustering, a cluster is defined as a set of data points where each point is closer to at least one other point in the cluster than to any point outside the cluster. The algorithm starts with a random data point and expands the cluster by including neighboring points within a specified radius.

Strict Radial Clustering, on the other hand, is a more restrictive version of Lax Radial Clustering. It considers only those points that have a minimum number of neighbors within the specified radius to be part of a cluster. This ensures that clusters are more compact and less likely to overlap.

K-means Clustering: K-means Clustering is a centroid-based clustering algorithm that partitions the data into K clusters. It assumes that the data points within each cluster are closer to the centroid of that cluster than to centroids of other clusters.

The algorithm starts by randomly selecting K centroids in the feature space. It then assigns each data point to the nearest centroid and recalculates the centroid based on the mean of the data points assigned to it. This process iteratively continues until the centroids converge or a stopping criterion is met.

K-means Clustering is widely used due to its simplicity and efficiency. However, it requires the user to specify the number of clusters (K) in advance, which can be a limitation. Selecting an appropriate value for K is crucial, as it directly affects the quality and interpretability of the clustering results.

2.3.2 Deep Reinforcement Learning

Deep Q-Learning is a reinforcement learning algorithm that combines Q-learning, a popular technique in reinforcement learning, with deep neural networks to handle complex and high-dimensional state spaces. It is a powerful approach for training agents to make decisions in environments with large and continuous state and action spaces.

In traditional Q-learning, a Q-table is used to store the values of each state-action pair, representing the expected future rewards for taking specific actions in specific states. However, for environments with high-dimensional state spaces, it becomes impractical or even infeasible to maintain a Q-table due to the exponential growth in the number of states.

Deep Q-Learning addresses this limitation by using a deep neural network, known as a Q-network, to approximate the Q-values. The Q-network takes the current state as input and outputs Q-values for all possible actions. It is trained to minimize the difference between predicted Q-values and the observed rewards obtained during interactions with the environment.

The training process in Deep Q-Learning involves an iterative approach. The agent interacts with the environment, selects actions based on an exploration-exploitation strategy (such as epsilon-greedy), and collects experience in the form of state-action-reward-next state tuples. These experiences are stored in a replay memory buffer, which allows the agent to learn from a diverse set of experiences and break the correlation between consecutive samples.

During the learning process, the agent samples a batch of experiences from the replay memory buffer and uses them to update the Q-network weights. The update is performed by minimizing the difference between the predicted Q-values and the target Q-values, which are calculated using a temporal difference (TD) target incorporating the observed rewards and the estimated future Q-values.

Deep Q-Learning has achieved remarkable success in various domains, including playing Atari games, controlling robotic systems, and optimizing complex decision-making problems. By leveraging the representation power of deep neural networks, it can learn directly from raw sensory inputs, making it suitable for high-dimensional state spaces.

However, Deep Q-Learning also faces challenges, such as instability and overestimation of Q-values. Techniques like target networks, experience replay, and double Q-learning have been proposed to address these issues and improve the stability and convergence of the algorithm.

2.3.3 Transfer Learning

Transfer learning is a machine learning technique that leverages knowledge gained from one task or domain to improve performance on another related task or domain. Rather than training a model from scratch on a new task, transfer learning allows the model to transfer and adapt the learned knowledge from a source task or domain to a target task or domain.

The idea behind transfer learning is that the knowledge and representations acquired by the model during training on a source task can be beneficial for learning a new task, especially when the target task has limited data or is significantly different from the source task. By utilizing transfer learning, models can generalize from previous experiences and expedite the learning process on the target task.

There are generally two main approaches to transfer learning:

Feature Extraction: In this approach, a pre-trained model, often trained on a large-scale dataset such as ImageNet for image-related tasks, is used as a fixed feature extractor. The early layers of the pre-trained model are frozen, and only the final layers or additional layers are added and trained on the target task-specific data. The pre-trained model has already learned general features from the source task, and these features can be used as input for the target task, effectively transferring knowledge.

Fine-tuning: In this approach, not only are the additional layers added on top of the pre-trained model, but the entire pre-trained model is also fine-tuned on the target task-specific data. By allowing the weights of the pre-trained model to be updated during training on the target task, the model can adapt and refine the learned representations to better fit the target task. Fine-tuning is particularly useful when the target task shares similarities with the source task, but some domain-specific adaptations are necessary.

Transfer learning can bring several advantages:

Reduced Data Requirements: By leveraging pre-trained models and transferring knowledge, transfer learning can alleviate the need for large amounts of labeled data for training a model on the target task. This is particularly valuable when data availability is limited or expensive to obtain.

Improved Generalization: Transfer learning allows models to learn more generalized representations by leveraging knowledge gained from the source task. This often leads

to improved performance on the target task, even when the target task has a smaller dataset.

Time and Resource Savings: Since the pre-trained model has already undergone extensive training on a large-scale dataset, transfer learning can significantly reduce the time and computational resources required to train a model from scratch on the target task.

Transfer learning has been successfully applied in various domains, including computer vision, natural language processing, and speech recognition. It has contributed to advancements in tasks such as image classification, object detection, sentiment analysis, and machine translation.

2.4 Network Slicing

Network slicing is a concept in telecommunications and networking where a physical network infrastructure is divided into multiple virtual networks, known as slices. Each network slice operates as an independent and isolated network with its own dedicated resources and specific characteristics. These slices are created to cater to different applications, industries, or user groups, allowing for customized services and optimized resource allocation.

Network slicing enables the efficient sharing of network resources while providing tailored connectivity, quality of service (QoS), and functionality to meet the diverse requirements of various applications or services. Each slice can have its own unique set of parameters, such as latency, bandwidth, security, reliability, and service-level agreements (SLAs), ensuring optimal performance for the specific use case it serves.

The implementation of network slicing typically involves virtualization technologies and software-defined networking (SDN) principles. Virtualization allows the physical infrastructure to be partitioned into multiple logical networks, while SDN provides the flexibility and control to manage and orchestrate the network slices dynamically.

By utilizing network slicing, service providers and enterprises can achieve several benefits. Firstly, it enables the creation of customized services that are tailored to the specific needs of applications or industries, such as enhanced mobile broadband, Internet of Things (IoT), or mission-critical communications. Secondly, network slicing allows for

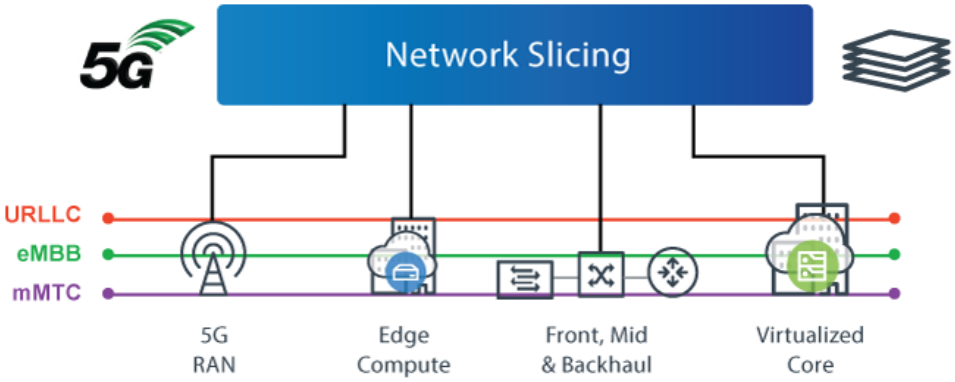

Figure 2.3: Network Slicing

efficient resource utilization, as each slice can dynamically allocate and scale resources based on demand, ensuring optimal resource usage. Additionally, network slicing offers flexibility and scalability, enabling rapid deployment of new services or applications and accommodating evolving needs and technologies.

Instantiating network slices involves several key steps:

Requirements Gathering: The first step is to identify the diverse requirements of different applications or services that will utilize the network slices. This includes factors such as latency, bandwidth, reliability, security, and specific service-level agreements (SLAs).

Slice Design and Mapping: Based on the gathered requirements, network operators design the architecture and configuration of each network slice. This involves defining the necessary network functions, resources, and quality of service (QoS) parameters. The design should align with the specific needs of the intended applications or user groups.

Resource Allocation and Isolation: Network resources, including computing power, bandwidth, and storage, are allocated to each network slice. Resource isolation mechanisms are implemented to ensure that one slice does not impact the performance or security of other slices. Techniques such as virtualization and software-defined networking (SDN) are commonly used for efficient resource management and isolation.

Service Orchestration and Management: Service orchestration platforms play a crucial role in managing and coordinating the lifecycle of network slices. These platforms

automate the provisioning, monitoring, and scaling of slices, ensuring efficient utilization of resources and dynamic adaptation to changing demands.

Security and Slice Isolation: Network slicing involves ensuring strong isolation between slices to maintain security and prevent unauthorized access or interference. Robust security measures, including encryption, authentication, and access control mechanisms, are implemented to protect each slice's data and operations.

Monitoring and Optimization: Continuous monitoring of network slice performance, resource usage, and QoS parameters is essential. Analytics and optimization techniques are applied to identify potential bottlenecks, optimize resource allocation, and improve the overall efficiency and performance of the network slices.

Benefits and Applications: Network slicing offers numerous benefits and finds applications in various domains:

Customized Services: Different slices can be tailored to specific applications, industries, or user groups, enabling customized services and functionalities. For example, slices can be optimized for enhanced mobile broadband, massive machine-type communications (IoT), or ultra-reliable low-latency communications (e.g., autonomous vehicles or critical infrastructure).

Resource Efficiency: Network slicing allows efficient sharing of network infrastructure among multiple services, maximizing resource utilization. It enables dynamic allocation and scaling of resources based on demand, ensuring optimal resource usage.

Flexibility and Scalability: Slices can be provisioned or decommissioned quickly to adapt to changing requirements or introduce new services. Network slicing provides flexibility and scalability to accommodate evolving needs and future technologies.

Service Innovation: With network slicing, service providers and enterprises can innovate and deploy new services or applications more rapidly. It provides a flexible and cost-effective platform for experimentation and service differentiation.

Chapter 3
Optimal Container Migration/Re-Instantiation in Hybrid Computing Environments

3.1 Introduction

The recent rapid adoption of 5G networks significantly increased the pre-existing interest in edge computing. This is mainly due to the 5G paradigm's readiness for rapid network changes coupled with its ability to accommodate the increasing number of users and generated traffic. Within 5G, edge computing allows providers to shift their services away from the core cloud and towards end-users by utilizing the abundant resources found in previously deployed under-utilized edge communication devices. This shift further lowers latencies and offloads traffic from the network's main backbone. Real-time dependent applications, such as 4K streaming, instant speech translation, and intelligent transportation systems may then run seamlessly [4, 5, 6]. The push to adopt edge computing has generated a significant amount of literature targeting the optimization of its main attributes with more focus on latency and energy costs, capital (CAPEX), and operational expenditure (OPEX) [7].

The concept of containers was proposed to further build on the premise of edge computing. Container-type solutions were introduced to replace traditional virtual machines (VMs) to increase the system's flexibility and scalability further. This change is due to the containers' ability to coexist in a shared platform, allowing more services to run with higher efficiency and mobility. Consequently, containers have lower CPU requirements compared to VMs that require hardware stack virtualization. Containers can also recover more quickly and require a significantly lower share of the system's memory without requiring an entire operating system (OS) image. These attributes allow for

microservices to be hosted on less sophisticated units. In contrast, robust computing units can be placed in the vicinity of end-users in the form of roadside units (RSU), base stations, and routers.

Containers hosted on the edge require additional constraints over core cloud-hosted containers to perform their intended tasks properly, as edge devices are typically less accessible and maintained less frequently. The containers must then achieve high robustness through other methods instead of relying on heavy hardware-based safeguards and traditional data center-based setups typically associated with core clouds [8]. This exposes the containers to more frequent outages, both planned (maintenance and updates) and unplanned (hardware failures and overloads) [9]. Redundant copies may be implemented to tackle this issue. However, this may affect the efficient utilization rates and system size, which are the advantages of using these containers. Thus, the desired approach is to maintain efficiency while lowering downtime when recovering, which lessens the impact on quality of service (QoS) once a failure is detected. This is vital in the edge environment due to the high availability requirement typical of its services, from urgent services, such as an emergency response, to standard services, such as IoT-based manufacturing and smart-city management. Traditionally, containers were allowed to either migrate from the failing edge node or re-instantiate by allowing the edge node to return to online status. Migrating with the anticipation of a failure serves to maintain their last state. The downtime will last until a new viable host is determined and the live capture image of the container is transferred. Conversely, the downtime in re-instantiation is dependent on the time a failed edge node needs to perform a hard reset and re-establish the connection with the end-user. This can be done by using the archived stateless image of the container instead of recovering the current data in the container.

An intelligent orchestration paradigm to decide between migration and re-instantiation is needed to achieve an optimal solution for placement while lowering downtime. Choosing the method depends on several network-related constraints to reduce global downtime. The decision process must also address the type of hosted services within the container and its compatibility and affinity towards migration and re-instantiation. For example, the migration should allow the system to recover seamlessly and maintain a high service up-time for a container with a stateful application. Comparatively, re-instantiation techniques are used when it is not necessary to preserve the application states [10]. The

recovery method is not controlled solely by the application's persistence requirement. Other metrics must be addressed, such as the end-user experience, cost, and security concerns.

Accordingly, this chapter introduces a novel container orchestrator based on an integer linear programming optimization model. The goal is to address the challenges of traditional orchestration and find the optimal placement with minimal downtime in hybrid computing environments. In addition, due to the time complexity of the optimization model, this chapter also introduces a comparably accurate heuristic model capable of achieving near-optimal results at a fraction of the time and complexity of the optimization model, thereby allowing it to provide real-time orchestration. The presented solutions are invoked once a failure is detected or deemed imminent. First, the host captures a snapshot of all hosted containers and generates a list of their given resources. It then provides a list of dependencies between the containers and similar constraints. A new placement is then generated using the decision, either migration or re-instantiation, to minimize the containers' downtime along with the access delay latency. To this end, the proposed approach enhances the QoS by minimizing the container downtime and satisfying the carrier-grade requirements of the provided services, namely availability and performance. The main contributions of this work are summarized as follows:

- Formulate the problem of container migration vs. re-instantiation while considering edge-related placement requirements such as latency and downtime.
- Develop an intuitive clustering method to generate representative user clusters capable of reducing solution space-related problems for optimization and heuristic-based solutions.
- Propose a real-time heuristic model orchestrator capable of providing comparable results to the optimization model.
- Evaluate the performance of the proposed optimization model and heuristic algorithm in comparison with the greedy migration and re-instantiation algorithms.

The remainder of this chapter is organized as follows: Section 3.2 presents some of the previous related work addressing this problem. Section 3.3 describes the system model adopted. Section 3.4 formulates the optimization problem. Section 3.5 presents

the proposed heuristic algorithm along with a brief discussion of its complexity. Section 3.6 presents and discusses the results of the system's performance evaluation process. Finally, Section 3.7 concludes the chapter.

3.2 Related Works

Current research trends describe a significant interest in exploiting the benefits of containers within an edge computing environment highlighting a number of critical aspects that require the attention of the research community [11]. A number of approaches were proposed to address the challenges facing the implementation of containers from both academia and industry.

Hawilo *et al.* created a solution focused on a specific type of VMs performing Network function virtualization (NFV) functionalities with the setup assuming all VMs are housed within a single data center [12]. The developed solution is based on an integer programming (IP) optimization model orchestrator. The system facilitates the placement of the virtual network functions (VNFs) taking into consideration different constraints such as inter-VM relations and service function chain (SFC) delays.

Barbalace *et al.* introduced a heterogeneous container migration scheme for natively-compiled containerized applications across compute nodes with differing instruction Set Architectures [13]. Their focus was on edge computing and migration schemes specifically to address the issues of stateful services. Their approach produces negligible overhead most noticeable during migration.

Tiago *et al.* proposed an analytical model to resolve Service Delay in edge cloud computing (ECC) systems. The approach seeks to minimize the delay for both communication and computation elements. The results were compared to models addressing processing delay only [14]. Although the authors tackle additional types of delays and have achieved an improvement over traditional methods, the lack of placement, downtime-related constraints, and delay can be considered a drawback.

Alam *et al.* leveraged lightweight virtualization to generate a modular solution that works with the Docker system [15]. Their solution achieved high availability metrics by relying on the innate redundancies generated by docker with their proposed solution allowing for on-demand service deployment on heterogeneous architecture layers.

Kuljeet *et al.* addressed the optimization problem by considering that the core cloud is optimized based on delay and energy with the decision whether or not to offload to the edge. The multi-algorithm service model operates by jointly allocating workload assignment based on assigned weights and computation capacities of the respective VMs. They also implemented a method to ensure the acquired results remain consistent [16]. While their approach is effective, it failed to take into account the network operation, failures, and containers' dependencies.

Abdullaziz *et al.* focused on the migration aspect of container orchestration by making it a more reliable option [17]. The authors achieved this by leveraging live orchestration as a method of achieving low downtime comparable to re-instantiation. Their method is broken down into two tiers. The first migrates user connectivity, while the second migrates user containerized applications. They have also addressed the possible causes of prolonged container migration downtime. Their results boast lower downtime by up to 50% shorter than that of the state-of-the-art migration.

Oleghe presented in his work [18] the frameworks and algorithms most commonly used in the container placement problem, the types of containers currently dominating the edge space, and the heuristic approaches favored in the research community to offer real time solutions.

Govindaraj *et al.* followed a similar approach in [19] with a focus on the complex orchestration of cyber-physical systems. Their work discussed the role of Edge Computing (EC) for factory automation applications. Taking Linux based containers as the basis, they built a live migration scheme called redundancy migration that reduced the downtime by a factor of 1.8 compared to the stock migration in Linux containers.

Feng *et al.* developed a unified mobile edge computing wireless power transfer (MEC-WPT) design framework with offloading and computing optimization for the edge while specifically relying on latency constrained computation. They minimized the total energy consumption subject to each node's individual computation and latency constraints [20]. The authors leveraged the Lagrange duality method to obtain the optimal solution in a semi-closed form.

Machen *et al.* took the idea of migrating containers and virtual machines in a layered format [1]. Their approach using readily available technologies starts by migrating the non-state related aspects of services or VM and achieved much lower downtimes than

traditional methods. The three-layer model also allowed the pre-caching of popular applications at MECs so that the time required for future instantiation of such applications can be shortened.

Mansoor *et al.* developed a classic linear assignment algorithm using computational nodes and presented a scheme for assigning VMs to data nodes that minimizes various latency metrics under different constraints [21]. The solution considered variable total access time allowances, with and without constraints, to showcase the system's adaptability.

Vaucher *et al.* developed a novel Container Orchestrator focusing on addressing issues such as having the containers being hosted on heterogeneous clusters [22]. Their focus was security-related issues stemming from the use of Intel-based software guard extension (SGX) enabled containers on a portion of the containers in a chain, and how to place them in such a way to maintain its relations to non-SGX enabled containers while preserving the security of the whole chain.

Manias *et al.* [23] developed a machine learning-based decision tree model for optimizing Virtual Network Function (VNF) placement. evaluating a few candidate ML models focusing on task offloading for network services. The most impactful model is the decision tree model, as it aids in the effective placement of VNF instances, thereby enhancing the overall performance and efficiency of service function chaining in network systems.

The Authors went on and expanded their work in [24] and introduced a novel DO-DAT algorithm, which focuses on minimizing the end-to-end delay while considering the depth of service function chains. Addressing the challenges in VNF deployment by effectively balancing the delay constraints with the network's topological considerations. Improving the efficiency of the network service delivery, particularly for delay-sensitive applications.

However, the solutions listed above focus on the recovery mechanism itself with focus on either re-instantiation or migration solely. In contrast, our work considers both potential recovery mechanisms as viable orchestration options. A second limitation of the related works is that their proposed solutions mostly exclusively consider the edge or core. This limits the optimality of the solutions to said solution space at a cost that could favor the users or providers at a time. Our solution considers the joint edge

and core orchestration, allowing for a larger and more comprehensive solution space. Therefore, this work proposes a comprehensive intelligent orchestrator based on an integer programming optimization model that aims to minimize the impact of migration and re-instantiation on the containers' downtime and access delay. Accordingly, the proposed intelligent orchestration frameworks follow a more comprehensive approach concerning the recovery mechanisms used, add the core to the solution space to act as an offset, and select the optimal placement that meets the container's demands, thereby achieving a highly optimized solution.

3.3 System Model

This work adopts a hybrid distributed computing environment similar to proposed works for modern networks. The global environment where the container and hosted services are placed is comprised of two platform types, namely the core cloud and the edge devices, as shown in Figure 3.1. The core clouds are typically hosted using server farms and data centers. This allows for an abundance of resources, both computing and memory, in these environments. However, the size of the required structures to host them, the infrastructure, and the human intervention required to maintain them greatly limit suitable geographical placements in the real world.

Conversely, edge devices are a newly tapped resource that became available in the wake of the adoption of software-defined networks (SDNs) and similar paradigms. That relies on the abstraction and virtualization of network functions to allow generic computing units to act as full-fledged networking units [25]. This presents a challenge for seeking an abundance of resources while also maintaining low latencies, which makes the optimization of this problem more crucial.

3.3.1 Physical Resources

The core clouds physically operate from data centers or server farms. The servers are given abundant resources compared to their edge counterparts. The servers communicate with servers housed within the same rack or separate racks within the same geological location. Cross-core communication is allowed due to the latencies, but it is not exercised extensively due to the increased complexity [26]. The latencies experienced

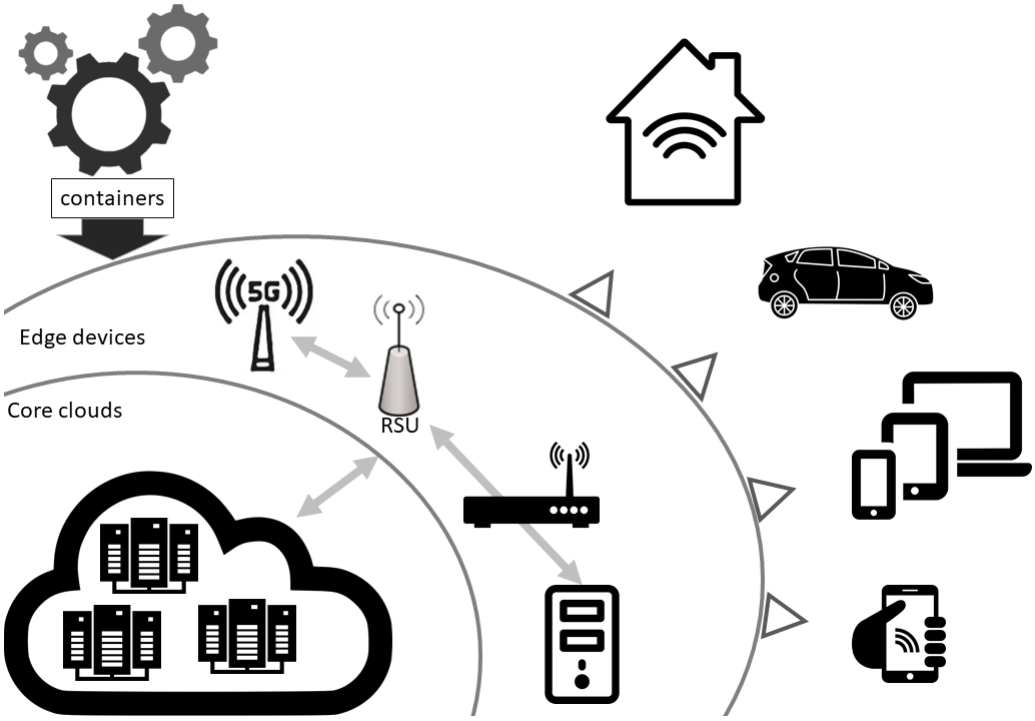

Figure 3.1: System Model: Core clouds and edge devices hosting service hosting containers

within the core are kept to a minimum due to the advanced communication backbone and high bandwidth mediums used between servers, typically fiber-optic cables. The latencies between the core cloud and user are much higher, mainly due to the propagation delay resulting from the distance between them. The amount of computing power and memory resources in each server are limited in variation due to the homogeneous nature of the physical rack servers typically used in data centers.

In contrast, the edge devices are typically either re-purposed communication nodes such as routers and RSUs, or larger units such as small dedicated edge nodes and 5G smart towers. The limited physical size of the edge units and the sporadic nature of their deployment enforces a singular rack structure. The racks typically host a limited number of servers compared to those found in large data centers. The latencies within the same rack are similar to those found in the core cloud. However, the delay between the edge device and end-user is much more negligible than that of the core cloud. The resources are more limited than those of the core cloud and have a higher variance between each edge node's memory and computation resources.

3.3.2 Containers

The containers and the hosted services can vary based on their latency requirements. This ranges from latency-stringent services, such as those related to financial transactions and security, to looser requirements, such as those related to text messaging and advertising services. In addition to the containers' latency requirements, the required resources are uniquely tailored based on the task they serve. For example, from high-memory, low-computation in trans-coding 4K videos and AR gaming to high-computation, low-memory requirements in navigation and sensor processing units.

3.4 Optimal Container Migration/Re-Instantiation (OC-MRI) Model Formulation

Container recovery combines two tasks with varying complexity. The decision on the method of recovery for a container between migration vs. re-instantiation followed by a placement optimization task resulting in an amalgamation that easily qualifies as NP-hard due to its inherent complexity and the exponential growth of possible solutions with the increase in problem size. The difficulty arises because there's a need to evaluate a vast number of possible configurations to determine the best solution. This evaluation process is computationally intensive and time-consuming; this can be mathematically represented as $O(M^N)$, where each of the N contain chains can be placed in any of the M servers. When dealing with large values of N and M, the number of possible placements increases exponentially, which makes exhaustive searches impractical. Due to this, finding the exact optimal solution within a reasonable time frame becomes unfeasible for larger instances, which is typical of NP-hard problems. These characteristics make a classical mathematical approach infeasible for live orchestration. However, we chose to develop it as a placement generator for benchmark purposes as well as its integral contribution to the robust development of supervised AI models in future iterations of the proposed solution.

3.4.1 General Model Description:

The model aims at minimizing two delay metrics. The first is the downtime resulting from the migration or re-instantiation process initiated for a container upon the imminent failure of its hosting computing node. The second is the access delay caused by the placement distance of the container to the end-user. The two processes have their own required static downtime, but choosing a new location has a significant impact on both the downtime and new access delay, as shown in Figures 3.2 and 3.3. The following section details how we optimally minimize the downtime within the objective function. Meanwhile, the access delay is optimally minimized by enforcing costs upon both the objective function and its related constraints.

- Computational resources constraint: Using this constraint, the proposed model selects the computing nodes that satisfy the containers' computational requirements. The resources are CPU cores and available memory.

- Network delay constraint: Using this constraint, the proposed model filters the nodes to select those that do not violate both the access delay and delay between master and slave containers in the cloud.

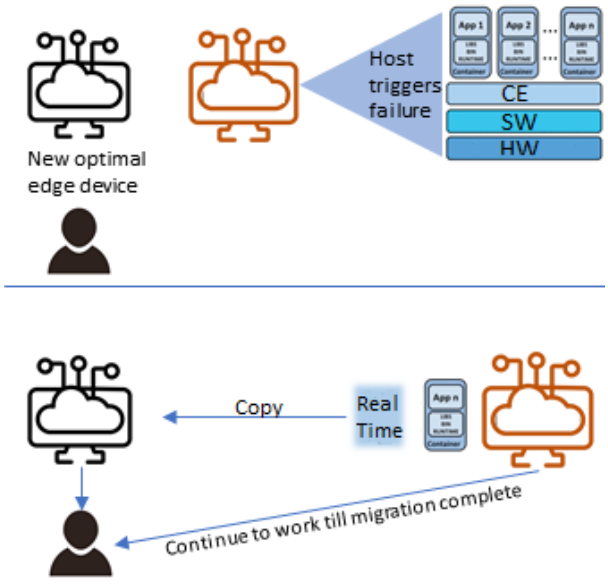

Figure 3.2: Migration mechanism with placement consideration

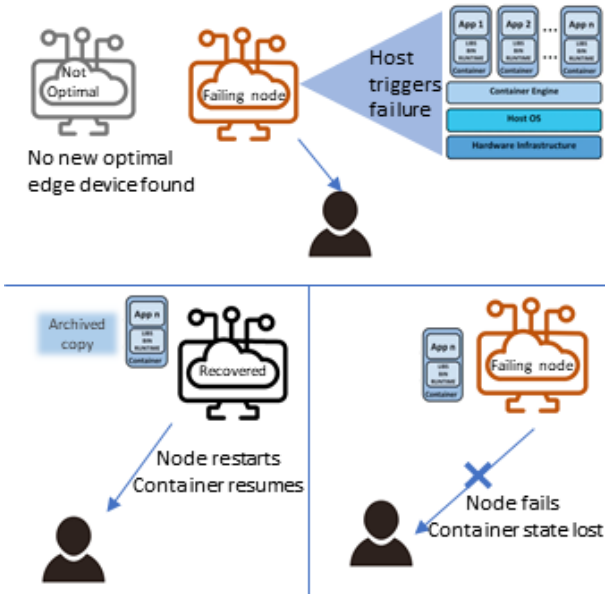

Figure 3.3: Re-instatiation mechanism with placement consideration

- Availability constraints: Each container is classified as unattached for single container-based services, master, or slave for services that consist of multiple containers working in a hierarchical set up to perform their tasks. To maintain the usability of QoS of the overall services offered by various containers, the proposed model defines the following constraints:

  - Affinity constraint: This ensures that the master container and its slaves must be hosted on the same physical server if the communication tolerance time between them is lower than the master's recovery time.
  - Anti-affinity constraint: Conversely, the slave containers and their master should be deployed on different servers if the slave has a higher tolerance time than their master's recovery time.

3.4.2 Notations and decision variables:

The developed model relies on the following set of variables as shown in Table 3.1. The table outlines each variable and how it relates to our problem setup covering all the types of constraints discussed in more detail below.

Table 3.1: Table of Notations

Variable	**Description**
X_{ce}	Placement decision variable
Y_c^{Dec}	Re-instantiation / Migration decision variables
C	Set of containers
N_c	Total number of containers
E	Set of host nodes
N_e	Total number of host nodes
U	Set of users
N_u	Total number of users.
Res_{cr}	Container c computational requirements
Res_{er}	Node e computational resources
R	Set of computational resources types
C^D	Set of slave containers
$X_{ce}^{original}$	Original placement of the container c
$T_{c'}^T$	Tolerance time slave container c'
T_c^R	Recovery time of master container c
SO_c^{Dec}	Container c hosting node's delay overhead
$AccD_{cu}$	Delay between end-user u and container c
$AccD_{eu}$	Delay between node e and end-user u
D_c^P	Delay generated from placement of container c
$D_{ce}^{ee'}$	Delay generated by moving container c from node e to e'
H_{mod}	Delay penalty based on containers' affinity to migrate or re-instantiate
D_c^{Dec}	Delay overhead of migration or re-instantiation decision

3.4.3 Mathematical formulation:

This subsection outlines the binary decision variables, the objective function, and the model's constraints.

- Decision Variables:

$$X_{ce} = \begin{cases} 1 & if \text{ Container } c \text{ is placed on node } e \\ 0 & \text{otherwise} \end{cases}$$

$$Y_c^{dec\_two} = \begin{cases} Y_c^{Mig} = 1 & if \text{ Container } c \text{ Migrated} \\ Y_c^{Mig} = 0 & \text{otherwise} \\ Y_c^{Re-inst} = 1 & if \text{ Container } c \text{ Re-instantiated} \\ Y_c^{Re-inst} = 0 & \text{otherwise} \end{cases}$$

- Objective Function:

$$min \sum_{c}^{N_c} DownTime_c + AccD_{cu} \tag{3.1}$$

- Boundary constraints:

$$X_{ce}, Y_c^{Dec} \in \{0,1\} \; \forall c \in C, e \in E \tag{3.2}$$

$$Dec \in \{Re - instantiation, Migration\}$$

$$Downtime_c \geq 0; \; \forall c \in C \tag{3.3}$$

- Placement constraints

$$\sum_{c=1}^{N_c} (X_{ce} \times Res_{cr} \leq Res_{er}); \forall e \in E, r \in R \tag{3.4}$$

$$\sum_{c=1}^{N_c} (X_{ce} \times AccD_{cu} \leq AccD_{eu}); \forall e \in E, u \in U \tag{3.5}$$

$$\sum_{E=1}^{N_e} X_{ce} = 1; \; \forall c \in C \tag{3.6}$$

- Availability constraints

$$(X_{ce} + X_{c'e}) \leq 2 \text{ or } \; (X_{ce} + X_{c'e}{}^{original}) \leq 2; \tag{3.7}$$

$$\forall e \in E, c \in C, c' \in C^D, T_{c'}^T \leq T_c^R$$

$$(X_{ce} + X_{c'e}) \leq 1 \text{ or } (X_{ce} + X_{c'e}{}^{original}) \leq 1; \tag{3.8}$$
$$\forall e \in E, c \in C, c' \in C^D, T^T_{c'} \geq T^R_c$$

- Re-instantiation/Migration delay constraints

$$Y_c^{\text{Re}-inst} + Y_c^{Mig} = 1;\ \forall \text{c} \in \text{C} \tag{3.9}$$

$$D_c^P = X_{ce} \times \left(\sum_{e=1}^{N_e} X_{ce}^{Original} \times D_{ce}^{ee'} \right);\ \forall c \in C \tag{3.10}$$

$$D_c^{Dec} = SO_c^{Dec} \times Y_c^{Dec};\ \forall c \in C \tag{3.11}$$

$$Downtime_c = D_c^{Dec} + D_c^P + H_{mod};\ \forall c \in C, \tag{3.12}$$
$$Dec \in \{Re - instantiaion, Migration\}$$

Constraint (3.2) determines that the placement and re-instantiation/migration decision variables are binary. Constraint (3.3) determines that the container downtime must be a positive number. Constraint (3.4) determines that the candidate computing node must meet the computational requirements of the potential container. Constraint (3.5) specifies that the access delay to the container is less than the threshold access delay of the corresponding container. Constraint (3.6) determines that only one computing node can host a container.

The containers in the solution space are classified as three types:

1. master: a container that relies on input from subordinate containers to finish its operation.

2. slave: a container or group of containers that are needed by others to operate but require no inputs for others to operate normally.

3. free containers: stand-alone single container services.

To maintain the interdependent relationship between different containers, constraints (3.7)-(3.8) determine if a container placement is dependent on its relation to its possible slaves. Thus, it either forces all related containers to a suitable physical location or allows them to be placed more freely. Furthermore, constraint (3.9) determines that a container can be either migrated or re-instantiated. Finally, constraints (3.10) and (3.11) determine that a container should be placed on a server that satisfies the delay requirements while minimizing the migration or re-instantiation overhead. Based on the previous constraints, the model selects to either migrate or re-instantiate each container to minimize its downtime. Lastly, constraint (3.12) shows that the downtime of each container is calculated in terms of the placement latency and the overhead delay resulting from the choice of recovery process offset by the H_{mod} modifier. This is based on the container's type, the critical nature of its state, or the information it holds controlling its affinity to re-instantiate or migrate.

3.4.4 Implementation

The proposed model is implemented using python environment for ease of use during experimentation. For the objective function in eq. (3.1), the values for both latency and downtime are generated based on the simulation environment created. At the same time, the resources-based constraints are predetermined based on the environment size and allocated edge devices. Finally, the generated matrices based on the container types are generated to include a portion with inter-dependencies to be used in constraints (3.7)-(3.8).

3.4.5 Complexity

Although integer programming problems can be solved using traditional branch and bound algorithms [27], these problems are typically classified as NP-complete [28]. This is further emphasized by the size of the problem's search space. Accordingly, the problem's search space can be estimated to be $2^{2\times N_c\times N_e}$ where N_c is the total number of containers and N_e is the total number of host nodes. This is due to the fact that there are 2^2 possible values (migrate or re-instantiate) for each container to be placed at each host node. For example, for the case of $N_c = 10$ and $N_e = 5$, the search space

is 1.267×10^{30}. Therefore, finding an optimal solution for this problem in a real-world scenario can be computationally infeasible due to the exponential growth in the search space size. Thus, a low-complexity heuristic algorithm capable of live decision-making is needed to address this problem.

3.5 Edge Computing-enabled Container Migration/Re-Instantiation (EC2-MRI)

The optimization model has allowed us to find the optimal placements through orchestration to achieve carrier-grade quality, whereby the mathematical model focused on satisfying different criteria, mainly downtime and latency. While successful at achieving optimal results, the model is computationally complex. This drawback is mainly due to its lack of scalability, as illustrated by the aforementioned complexity analysis. Without real-time orchestration, it is counter-intuitive to offer the system in its current state for orchestrating highly dynamic environments, such as mobile edge computing due to the following:

- The need for total access to the solution space for proper orchestration.
- Treatment and optimization of users and containers as individual unique elements.
- High computational time and significant resources required to perform the orchestration task.

Given that downtime is the primary optimization objective, the model's orchestration processing time must be taken into consideration. This additional delay is especially significant when considering the highly mobile nature of edge users, and the sporadic nature of SDN and 5G based edge devices where it may become detrimental. The optimization model cannot be retrofitted or adjusted in a way that would address all the aforementioned issues while maintaining its accuracy. A better lightweight real-time solution is thus required. We choose to pursue a heuristic-based approach due to its flexible nature and capability of handling frequent changes to the network. They can also circumvent the global environment's staggering size and its effect on any system's ability to provide seamless orchestration unnoticeable to the end-user. Orchestration can benefit

from taking place within the edge environment by removing the communication overhead of offsite orchestration. This design aspect minimizes the additional delay caused to the orchestration communication overhead by a core cloud that is typically required when done using the centralized approach. This requirement is addressed by limiting the heuristic model's complexity, thus allowing it to run on the edge environments, eliminating the drawback. However, to achieve this, the overall solution space must be segmented into isolated sub-spaces to maintain the required low complexity.

The requirements above present their own set of challenges when creating the heuristic model. To address them efficiently, the system's behavior is comprised of two main stages. The first intends to run offline and sporadically, and focuses on simplifying the solution space by tackling the number of elements through clustering and chaining. It also further simplifies the problem by segmenting the generated clusters and chains based on their interactions and proximity to each other. This stage is intended only as an environment initialization point. It is a fail-safe when the existing edge-space results in many failures that indicate a considerable divergence from the last segmentation cycle. Its offline nature gives it access to core cloud environments' exclusive resources, especially the abundance of low-cost computational power and access to the global solution space. Figure 3.4 shows the breakdown of the undertaken task. Firstly, the generation stage is where the solution space is simplified, followed by the auditing stage where the

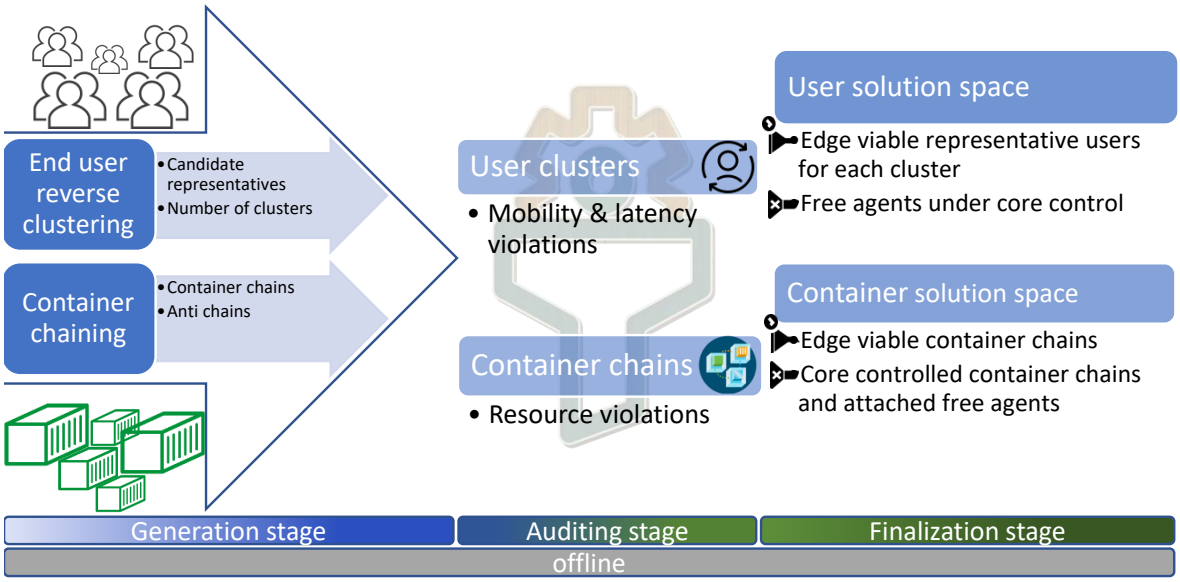

Figure 3.4: User and container solution space setup

candidates are vetted for accuracy and reliability. Lastly, the finalization stage is where the heuristic assigns control of the outputted elements to the best-suited orchestration controller, as discussed below.

3.5.1 Generation Stage

The live requirement placed on the edge-computing container migration/re-instantiation (EC2-MRI) model demands limited complexity. However, relying on a greedy or similarly generic approaches, while boasting the highly sought-after simplicity, not only do they not guarantee near-optimal placements, their output can vary vastly. To achieve our target of matching the placement benefits of the OC-MRI model, we must first solve the solution space size issue. This solution can be achieved by reducing and segmenting the solution space into isolated and self-managing subspaces.

3.5.1.1 User clustering

The OC-MRI is allotted several concessions regarding the edge user's main distinguishing attributes since it was not developed as a live orchestrator, with the main concession being its ability to ignore the mobility of user clusters. However, the heuristic approach cannot follow suit. As such, the process of generating the user solution space must be given additional attention. This is to ensure that the system is generating a healthy and stable solution space and not fall into a counterproductive cycle of "build-to-fail" environments. This outcome will keep triggering the offline stage more frequently, with little actual orchestration taking place. When generating a new edge space, simply permitting all users entry into the space as individual entities lacks scalability and reproduces the OC-MRI's drawbacks in our heuristic solution. On the other extreme, using advanced clustering techniques can be too time-intensive even for offline operations. Their high granularity will also steer the solution back towards the build-to-fail state. We address these hurdles in an intuitive approach by first subjecting the user solution space to the subtractive clustering process. This process generates the clusters necessary to provide coverage for the overall solution space and a corresponding list of representative candidate users acting as the centroid of their host cluster. While not highly complex, this clustering method has been implemented several times and proven

to work well. It was used most recently in the research efforts within 5G ultra-dense networks [29]. The main premise is segmenting user space to ensure the distance between the candidate clusters is significant by raising the Squash factor related to the overall user space and adjusting the Accept/Reject ratios based on the average mobility index of the users occupying each edge space.

The results of the subtractive clustering process are a number of disjoint clusters. From said list, we choose a candidate that is near the center of the generated segment to act as the representative user for the purposes of generating the audit threshold value and the mobility audit as well. After the centroid list is generated, we apply a straightforward rigid distance-based clustering that produces a radial border around the centroid. This captures as many users adhering to equation (3.13) where S is the user's diameter coordinate space divided over the number of potential clusters. T is an integer variable ranging from 1 to 3 representing the mobility of each user (stationary, pedestrian, vehicular) and I represents the tolerances of the users present in the overall user space. This equation is used to limit the cluster upper limits in both the physical coverage area and the number of users hosted. It also nearly eliminates the presence of build-to-fail user clusters scenario.

$$Max_{\text{clusteringdistance}} = (ST^{\text{unclustered}} - SI_{\text{min}}) \tag{3.13}$$

3.5.1.2 Container chaining

The optimization model allowed containers with links to exist on multiple edge devices given the tolerances outlined in equation (3.7). The heuristic model cannot handle such complex tasks as the placements and interactions can easily grow in complexity quite rapidly. To address this issue, a reduction step similar to what was done in the user solution space is implemented. Unlike the user solution space, segmenting and clustering are not required in their traditional meaning. Instead, containers are chained together based on their affinities as shown in Algorithm 1.

3.5.1.3 Chain generator

The chain generator aims at maintaining the size and complexity requirements of the second algorithm. Individually orchestrating the containers will be difficult. Thus,

the algorithm generates a master list of all container chains based on the affinity constraint (3.7). Container chains are then clustered into singular entities and assigned new IDs. The new format changes the IDs from integers to floating variables with the integer digits representing the cluster ID and the fraction digits preserving the container unique IDs. Anti-affinity is then generated based on constraint (3.8). Once the key elements of the table are generated, the corresponding computational requirements for each container chain is amended based on the aggregated value of contained clusters. Lastly, an affinity binary variable is generated based on the highest H_{mod} modifier for each cluster chain. While this approach guarantees a semi-optimal solution, it is still limited by the loss of multi-host based solutions due to the over simplification of the affinity constraint to maintain a lower complexity in the latter algorithm.

Algorithm 1 Chain Generator

Input: $C = \{1, 2, .., |Nc|\}$, CD, Segment tables
Output: Cluster & Anti Cluster tables
for $e \in Clustertables(i)$ **do**
  for $c \in Nc$ **do**
    if $(X_{ce} + X_{c'e}) \leq 2$ *or* $(X_{ce} + X_{c'e}{}^{original}) \leq 2$ **then**
      Rename c to $ChainID.c$
      update $CnChainTable(u(i))$
    end if
    if $(X_{ce} + X_{c'e}) \leq 1$ *or* $(X_{ce} + X_{c'e}{}^{original}) \leq 1$ **then**
      update $AntiCnChainTable(u(i))$
    end if
  end for
end for
return Cluster & Anti Cluster tables

3.5.2 Auditing stage

Once the users have been placed into clusters and the containers have become grouped into more monolithic chains, we have to ensure the feasibility of our new solution space in various aspects. We begin addressing this with the user clusters. Two major causes of failure have to be avoided. First the tolerance offset from the representative user violates that of the masked user (cluster host user). The second is a preventative step to avoid immediate failures related to mobility of the users. To address this issue,

an auditing threshold is first enforced on each cluster guided by the following equation (3.14) where I is the mobility index of the user and i is the cluster number:

$$Audit_{\text{threshold}} = (Max^{i}{}_{\text{clusteringdistance}}/2) - I \tag{3.14}$$

Once the auditing threshold is invoked, all users found in violation of it, based on their tolerances or mobility, will be checked to ensure that they can remain members of the cluster and the representative user can be an effective stable substitute. Failed users are pulled from their host cluster and designated as a free agent under the control of the core until the next clustering cycle. The next cycle is triggered periodically or when triggered based on significant degradation in the EC2-MRI performance. The degradation can be caused by significant changes in the host devices' number and available resources or the live portion of the auditing stage offloading a more extensive than allowed number of users from the clusters to become free agents under the direct core cloud control. Figure 3.5 illustrates the generation of a threshold and an example of a mobility violation-based user expulsion.

Container auditing on the other hand is relatively more straightforward. By using the overall size of the container chain, it checks to ensure that the available containers are placeable while maintaining the latency constraints. In addition, to maintain feasibility through multiple iterations, a resource-specific over-provisioning is enforced. The margin limits the maximum resources for a container and removing a whole chain ensures that the remaining chains remain hostable. This approach is necessary given the relative opportunistic nature of the used placement algorithm and to avoid build-to-fail states. Any chains found to be violating this threshold will be removed along with their attached users from the clusters.

3.5.2.1 Edge device allocation stage

Once the auditing stage is finalized, the edge devices that can house the remaining cluster chains are polled sequentially and an ordered candidate list is prepared based on Algorithm 2.

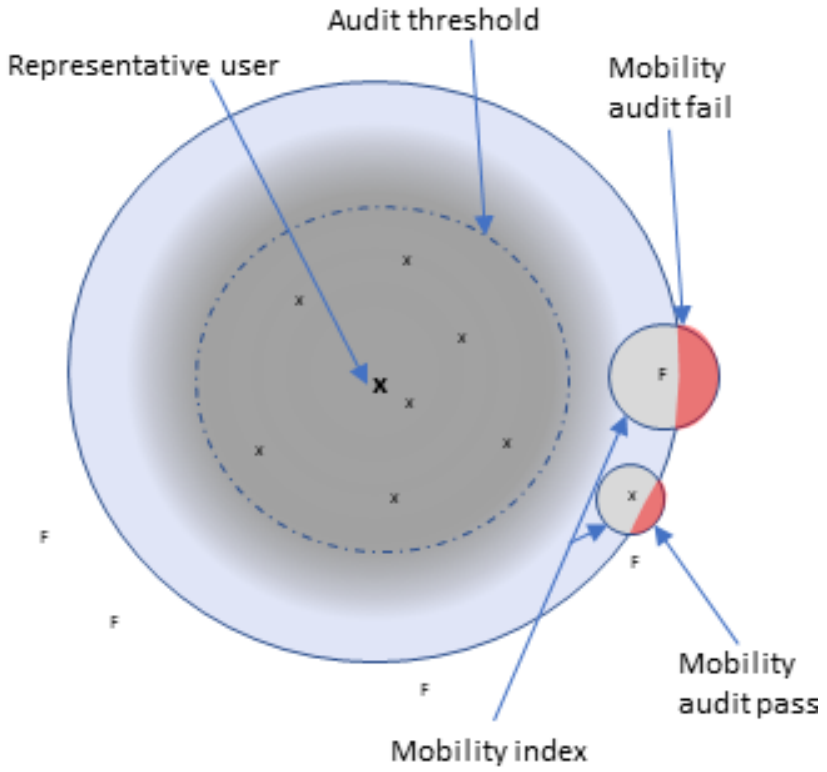

Figure 3.5: Auditing stage, the red portion highlights the margin of mobility violation

Algorithm 2 Segmenter

Input: $E = \{1, 2, .., |Ne|\}$, $U = \{1, 2, .., |Nu|\}$

Output: Segment tables

2: **for** $u \in Nu$ **do**
3: **for** $e \in Ne$ **do**
4: **if** $Dp(e(i)) < UserCluster(MaxTolerance)$ **then**
5: **update** $SegmentTable(u(i))$
6: **end if**
7: **end for**
8: **Sort** $SegmentTable(u)$ based on latency to $u(i)$
9: **end for**
10: **return** Sorted Segment Tables

Segmenter: Generates a single dynamically sized table of candidate edge devices

based on their availability within a user latency allowance. The table size is based on the highest latency threshold. This can result in a significantly large table in cases of densely placed servers coupled with loose latency requirements, which is addressed in step 4. The chain generator is polled for each region after all the tables have been generated to eliminate any edge devices deemed incapable of containing the generated container chains. The servers are then organized based on latency and computing power. Then, only the top candidates for each region are maintained and the rest are discarded. The number of top candidates maintained is based on the density of edge devices found in each region.

3.5.3 Edge-Controlled Live Orchestration Placement Stage

The model is trigger-activated once a container or VM signals a failure or fails to respond, enacting Algorithm 3. Starting from the hosting edge node, an iterative check of the candidate table is initiated. This is followed by sequential requests based on their ranking. The requested edge devices check their computational availability and their currently hosted containers for any anti-affinity violations. If none are found, the request is approved and the algorithm terminates. If the table is exhausted with no solutions, the edge device raises the request to the core hosting secondary algorithm to orchestrate a viable solution. At this stage, the drawbacks of the clusters in the first stage are rectified, but the container chain is removed from the candidates list for that region until the next periodic operation of the provisioning stage.

3.5.4 Core-controlled Live Orchestration Placement Stage

The heuristic chooses to treat a container differently if it is a free agent from the start or was removed due to repeated failures. This method maintains overall system complexity. However, relying on a generic approach, such as a greedy migration or re-instantiation, would be deemed counter-intuitive because it can keep victimizing a group of users to an unhealthy edge device or a sub-optimal latency. To address this, the core can make use of its abundance of resources to orchestrate such stragglers using the original methods outlined in the OC-MRI model. The main limitation of the optimization model was the lack of scalability with relation to the solution space. However, while acting as

Algorithm 3 Orchestrator

Input: Cluster & Anti Cluster tables, Segment tables, $ChainID.c$
Output: placement request
3: **for** $i =< sortedsigmenttablesize$ **do**
4: **while** $X_{s_u} \neq 1$ **do**
5: **find** $d^c_{s_u} = \min_{c \in C_u} \{\frac{1}{|V|} \sum_{v \in V} d^c_{s,v}\}$
6: **if** $ChainID \neq AntiClustertable(i)$ **then**
7: **if** candidate resource allowance $> ChainID$ resource requirments **then**
8: **poll** candidate
9: **if** placement successful **then**
10: **EXIT**
11: **end if**
12: **end if**
13: **end if**
14: **end while**
15: **end for**
16: **if** no candidate found **then**
17: **remove** $ChainID.c$ from Cluster, Anti Cluster tables, and Segment tables.
orchestration raised to core cloud controller
18: **end if**
19: **return**

a backup to the EC2-MRI model, this is no longer a requirement. When orchestrating has either failed during the live stage or in the case of the free agents, the core can use the distance S to create the solution space limits for a single user and perform live orchestration. The core can tackle this orchestration in two methods based on the number of entities under its control at the instance of a failure:

1. Use the free agent's tolerance to generate the solution space.

2. Use the distance S to generate a slightly complex solution space and optimize all controlled nodes within the influenced region.

3.5.5 Complexity

In contrast to the OC-MRI model, the proposed EC2-MRI heuristic has taken a number of steps to avoid such a limitation and focused on achieving a lower computational complexity, even with the discussed two stages. The overall complexity is governed by the complexity of each of the stages. The complexity of the first stage being $O(N_u \times$

$N_e + N_c \times N_e$) where N_u is the number of users, N_e is the number of host nodes, and N_c is the number of containers to be placed. However, since this is done only once offline, this will not impact the system's overall complexity. On the other hand, the complexity of the second stage is $O(N_c \times N_e)$ under the assumption of the worst case being that each container is placed freely with zero affinity. Thus, the overall complexity is linear in the number of containers and host nodes. Using the same values as in the previous example, the complexity would be in the order of 50 operations.

3.6 Performance evaluation

To best test the two solutions put forth, a suitable device is necessary for both bench-marking and environment simulation. To evaluate the performance of the proposed solutions, a physical workstation with 6 Cores and 12 threads of CPU, a 11 GB GPU, and 32 GB of RAM is used to build the simulation environment and implement both the OC-MRI and EC2-MRI models.

Three simulation environments of varying size are implemented to represent settings from sporadically populated to densely populated edge environments and to better represent and highlight the variable nature of edge environments. Once the simulation environments are properly populated, both models are tested in contrast to two generic algorithms, namely greedy re-instantiation and migration. These greedy algorithms are introduced to act as base benchmarks to highlight each of the proposed models' benefits and drawbacks. The algorithms' behavior is created through assigning a higher affinity to either the migration or the re-instantiation decision variables to force the intended behavior without allowing for non-solvable states. The resulting latency and downtime generated by each orchestration approach is measured from the users' point-of-view to act as the main metrics for the models' performance. In addition, other metrics specific to the clustering algorithm are presented to highlight the efficacy of the proposed clustering approach. Finally, given that the EC2-MRI is the only implantable solution, further discussion and metrics related to its inner modules are presented.

3.6.1 simulation environment

A number of simulation environments are generated to perform the most comprehensive method of testing the two models and the bench markers, with equal distribution to each size as shown in Table 3.2. Accordingly, the simulation environment is generated through five stages starting with the user clusters, core placement, edge devices, resource allocation, and finally container placement. Before any testing takes place, the earlier seeding of the users and containers gets remapped based on their cross latencies where the distance between any related entities such as core cloud and edge device is represented as distance. The only caveat is for any communication taking place over the core-edge separation region, typically between core cloud and edge devices or end-users, will incur a flat latency penalty to represent the typical lag of traditional network backbone, which we are trying to minimize.

Table 3.2: simulation environment size

	edge	core	user clusters
small	15	2	4
medium	55	3	12
large	125	3	21

3.6.1.1 User Cluster Seeding

The EC2-MRI offline stage's related processing delay is not taken into account to maintain objectivity, as the users' clusters assumed to be present in the OC-MRI are used from the output of the EC2-MRI finalization stage. Additionally, all processing time for the models left are ignored when measuring the downtime to focus on the orchestration effect alone. However, the heuristic failed call and response attempts are included as they are not deemed to be related to processing and are an integral part of the heuristic model's behavior.

The user clusters are generated based on the user clustering algorithm outlined in Section 3.5.1.1 of the heuristic model. Each cluster is occupied with a minimum of five users and capped at 25 users. To ensure proper clustering, a randomized mobility metric is attached to the users while ensuring that no highly mobile users are also given

stringent latency requirements. This is done to maintain the solvability of the solution space for all algorithms and maintain the practicality of the simulation environment. Once the clustering and auditing stages are completed, separate sub spaces are generated around each cluster. As shown in Figure 3.6, the outline of core edge latency barrier is highlighted along with the representation of the user clusters and free agents. At this stage, the only restriction aside from only seeding in the edge space is the overlapping between user clusters, where the smaller clusters are quashed and made into free agents. This step is enforced to ensure that no unrealistic latencies occur in the later stages of simulation environment setup given that the edge devices are generated related to the cluster centroid.

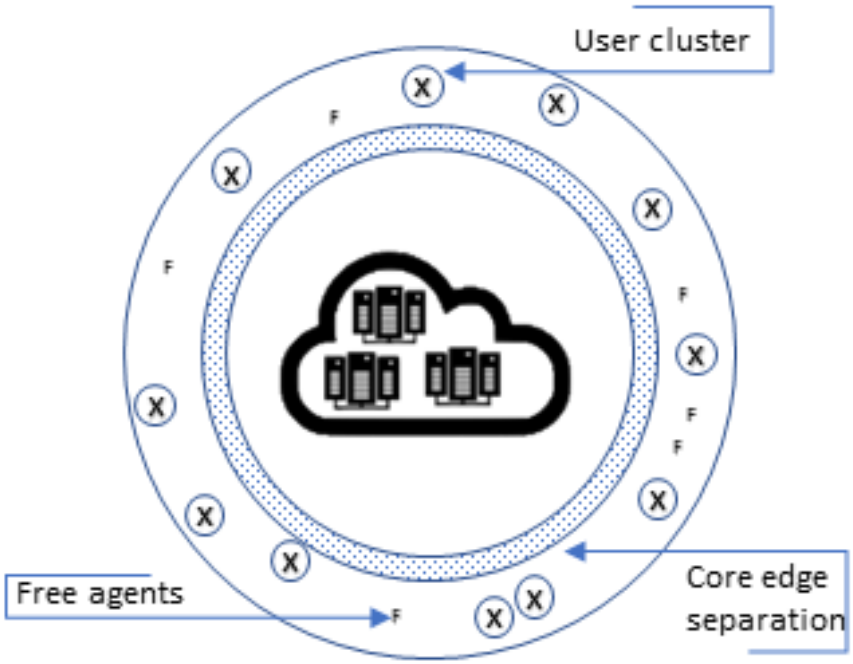

Figure 3.6: User clusters seeding and core edge latency penalty

3.6.1.2 Core Cloud Seeding

The core clouds are first assigned a location within the core designated space. Following this, a number of servers will be assigned to each of the cores to represent tray workstation style devices with minimal latencies. Once the servers are assigned to a distinct core, they are then designated to their specific racks within each core. This generates the necessary latencies within the data centers and between the core clouds and user clusters. The delays generated by their placement within the core are not represented in the latency map. But to maintain accuracy, a latency for inter-rack communication is randomly assigned in the range of 2-5ms, and 7-10ms for cross-rack communication.

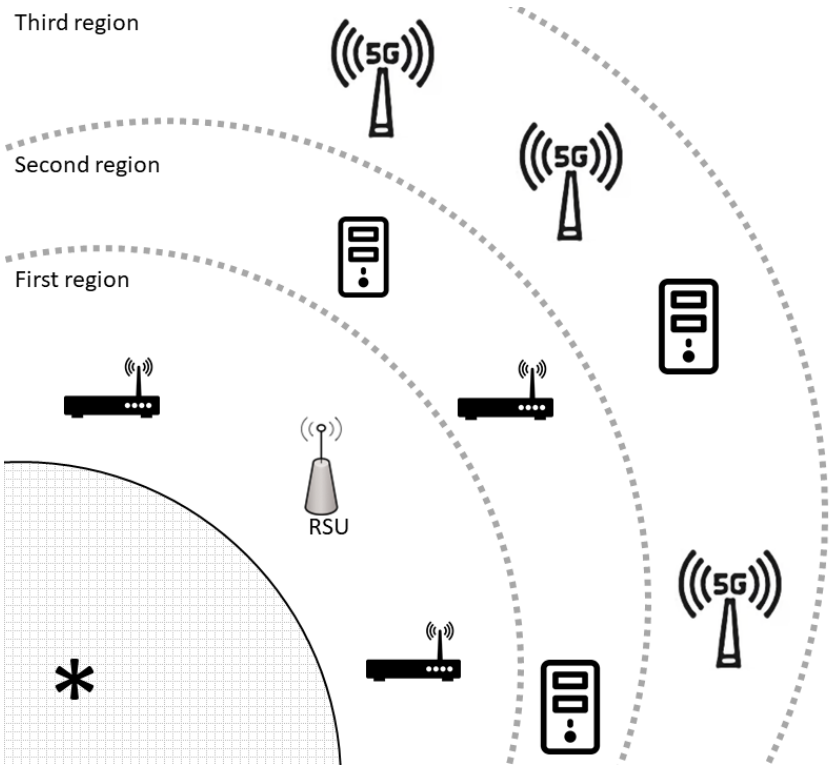

Figure 3.7: Edge device placement with respect to user cluster

3.6.1.3 Edge Device Seeding

The edge devices' placement in the vicinity of the user clusters impacts both the possible size of the edge device and its offered resources. The sizes of used edge devices fall into three generalized tiers: small for devices such as routers, medium representing devices such as an RSU, and large for devices similar in size and capability to 5G towers.

The probability of these placements has been controlled as shown in Figure 3.7. The regions are split based on their latency with respect to the user clusters. The mapping-based latency uses a scale from 10ms to 32ms representing typical edge latencies within levels 2-4 of 5G environments as outlined in [30]. The distribution for each edge device type is static in all simulation environments generated. The ratios for each used type as shown in Table 3.3 outline each region's ratio for total device types from the main pool based on the assigned percentage.

Table 3.3: Edge device size-based placement

	Region		
Edge Device Size	First	Second	Third
Small	70%	20%	0%
Medium	30%	65%	35%
Large	0%	15%	65%

3.6.1.4 Resource Allocation

The available resources for services are categorized as either computational power or memory. The edge devices will have an abundance of either but not both. The unit of measure for both will be in their capacity to meet the demands of a singular small container of either type. Table 3.4 shows the allocation based on the size and type.

Table 3.4: Edge node resource Availability

	Computation affinity		Memory affinity	
	CPU	Memory	CPU	Memory
Small	4	2	2	5
Medium	9	5	5	9
Large	15	7	8	13
Core	18	12	18	12

3.6.1.5 Container Placement

The containers are categorized through a similar method to the resource allocation based on the size requirements and their resource affinity. Table 3.5 shows the relation

between their size and the required resources. However, they additionally require a latency threshold. The containers are assigned affinities to represent container chains performing a singular service where cross communication is required, then the latencies between them are restricted. Once the containers are randomly distributed between user clusters, the downtimes assigned to each container relates to its type and size with values ranging from 2s to 5s with additional time tacked on GPU heavy containers following the approach in testing used in [31]. The aforementioned affinities and overall latency requirements are adjusted based on the length of the chain to maintain both solvability and plausibility. Finally, a 15% anti affinity is assigned to any container not belonging to the same chain. These probabilities are maintained throughout the general testing.

Table 3.5: Container resource requirements

	Computing affinity		Memory affinity	
	CPU	Memory	CPU	Memory
Small	2	1	1	2
Medium	4	2	1	4
Large	7	4	2	6

3.6.2 Downtime and Latency analysis

Both the latency and downtime are generated by the placement distance in the latency mapping during edge device seeding stage. The ranges used stem from typical downtimes expected with general current memory based migration techniques [32][33]. The downtime used as input for the models tests relies on the type of container recovering along with the transfer time associated with the typical size of said containers.

3.6.2.1 Downtime

The proposed optimization and heuristic models are bench-marked against two base greedy algorithms. The models experience a highly varied downtime response when transitioning between the small towards the large simulation environments. The resulting range of downtime experienced has been divided into 10 segments of equal length to better contrast the different models' behavior within each time range. The OC-MRI, as is shown

in Figure 3.8, performs best in the sporadic environments with high adherence to the effect of the container type modifier H_{mod}, with a majority of placements falling within the 3.25 to 4.75 seconds range. The optimization model maintained great performance throughout with minimal additional lag even when expanding to the medium and large simulation environments, peaking at 4 and 5.5 seconds, respectively.

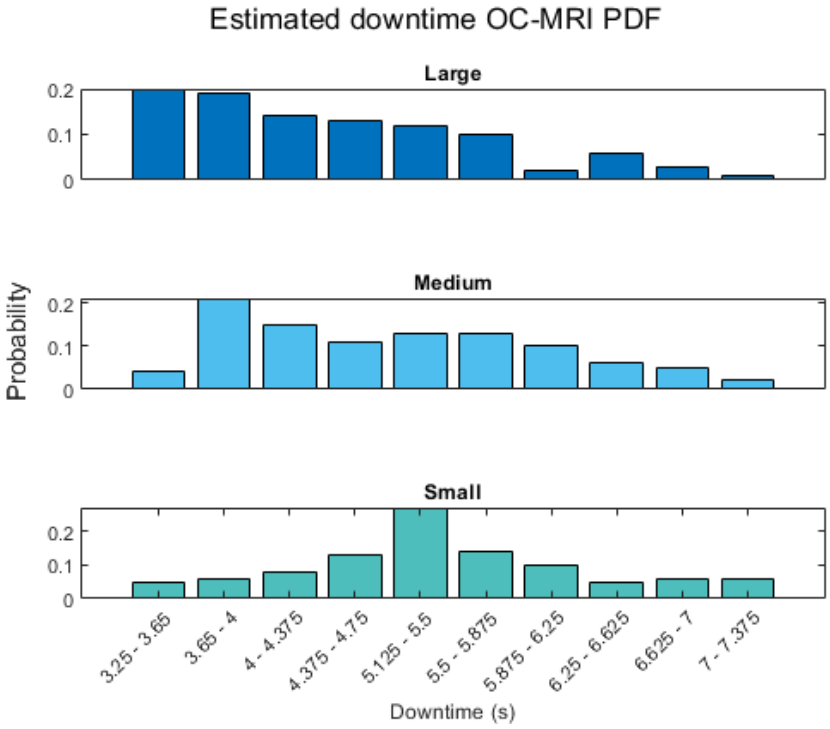

Figure 3.8: Downtime distribution OC-MRI

Following the performance of OC-MRI, the heuristic based approach used in EC2-MRI performed comparably well with respect to downtime as shown in Fig. 3.9 boasting similar results. However, it is noteworthy that it experiences a consistent dip in the placement opportunities within the initial 2.5 second range. This is attributed to two factors, the monolithic chains presence making the use of the low latency of small edge devices less likely due to their limited resources, and the presence of conflicting modifier H_{mod} values within a single chain both leading to higher latencies. Finally, the performance of the base bench marking algorithms has resulted in a much higher downtimes with very limited placements in the first 1.5 seconds totaling less than 25% for all simulation environment sizes combined in each algorithm as shown in Figures 3.10 and 3.11.

Overall, the OC-MRI model maintained the lowest downtime values throughout the

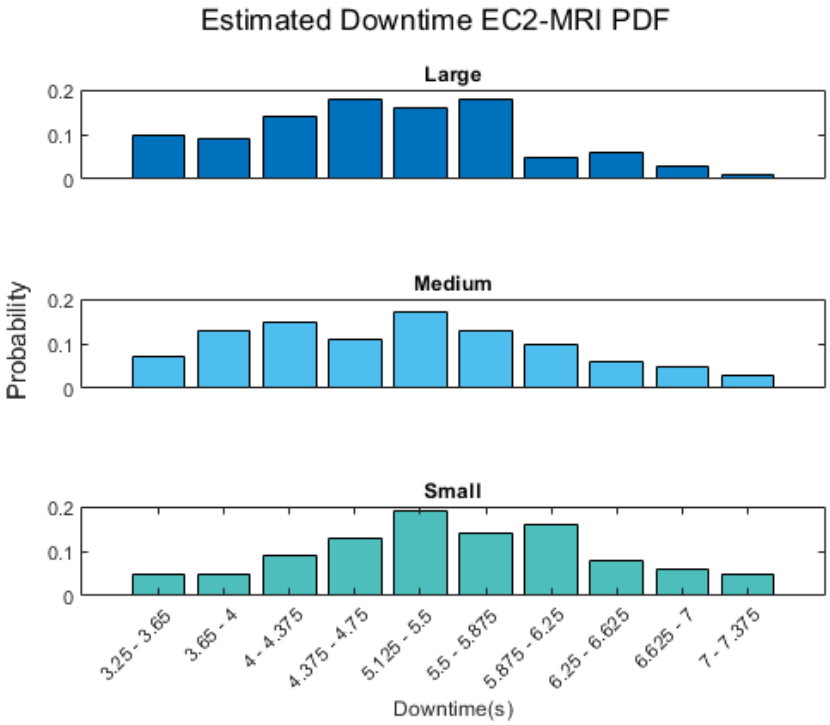

Figure 3.9: Downtime distribution EC2-MRI

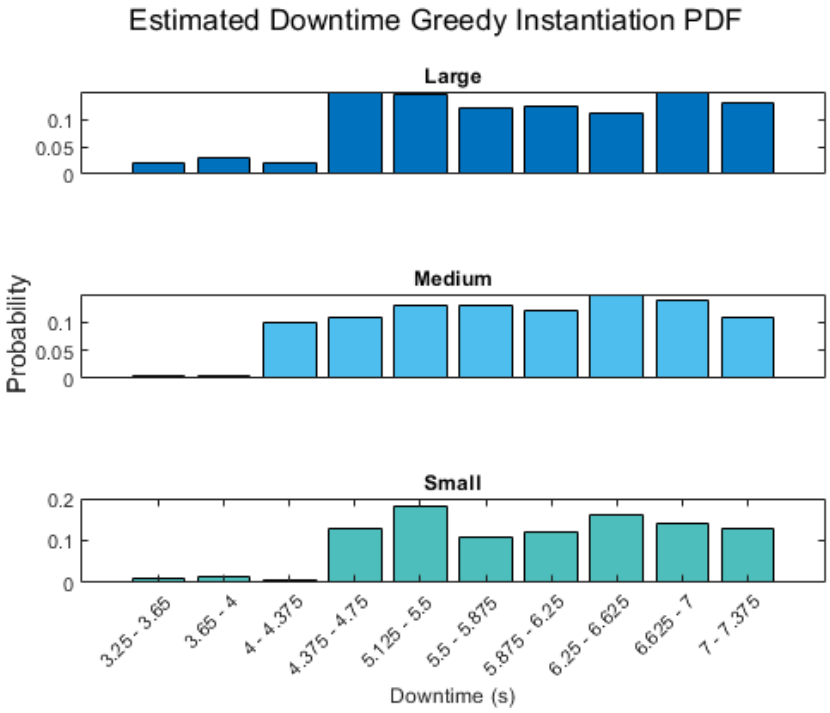

Figure 3.10: Downtime distribution greedy re-instantiation

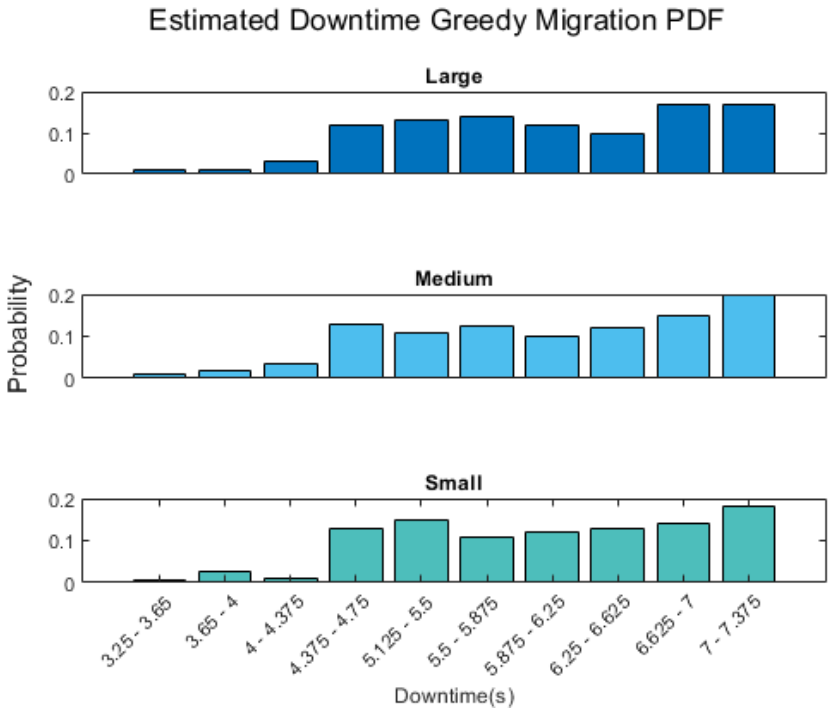

Figure 3.11: Downtime distribution greedy migration as per [1]

latencies with high adherence rate to the H_{mod}. The heuristic approach, while showing a few drawbacks related to container chaining, still maintained comparable downtime values to the optimization model. The greedy algorithms using the H_{mod} modifier were turned into a forced migration orchestration modeling the behavior similar to the recovery method in [31] had significantly higher downtime regardless of the effect of the container type. This is due to their simplistic approach, ignoring the container requirements, and relying solely on their pre-configured biases.

3.6.2.2 Latency

Similar to downtime, the minimum and maximum latencies generated by all models were used to create 10 time-based segments to allow for better contrast when comparing the models' behavior to each other with each latency range. The latencies offered by the OC-MRI followed a similar trend to its performance in the downtime metric with great distribution heavily weighted towards lower latencies as shown in Figure 3.12. While the performance between the small and medium simulation environments was comparable, a significant lag was generated during the large simulation environment bench-marking

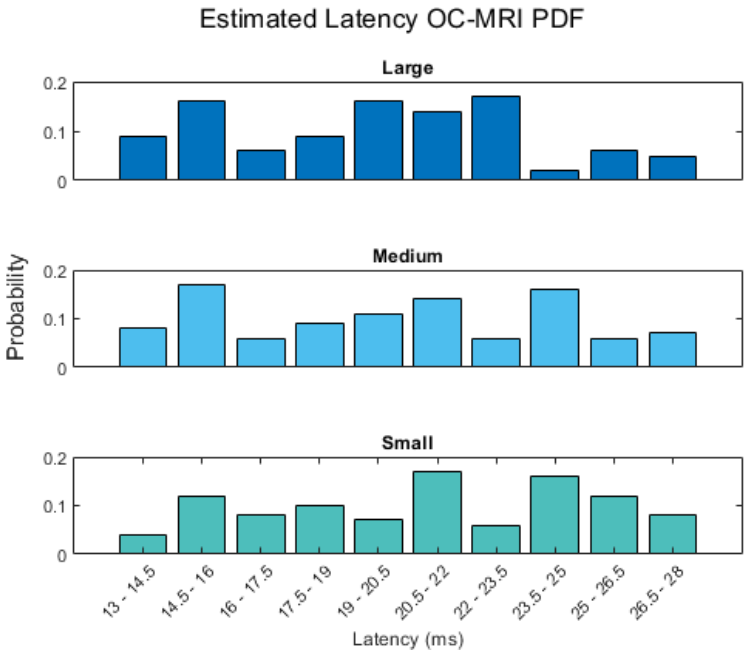

Figure 3.12: Latency experienced by the user, OC-MRI.

with a noticeable reduction in the placement opportunities in the lower latency thresholds. The performance of the EC2-MRI favored the middle range of latencies with a slight offset towards the upper range in the large simulation environment stage as shown in Figure 3.13. This can be attributed to the necessary shift away from the user caused by the container chaining favoring the middle and large sized edge computing units.

Both base benchmark algorithms have shown inconsistent latency response to the increase in the simulation environment size. The results of the greedy recovery mechanisms were heavily weighted towards the upper 4ms with little to no change in the distribution throughout the testing process as shown in Figures 3.14 and 3.15. The distribution notices a failure to allocate proper placements when approaching the outer region of the edge space adjacent to the core with the H_{mod} variable impact on their operation becoming more negligible. While the EC2-MRI has consistently achieved comparable results to the OC-MRI optimal placements, the cost of reducing the size of the solutions space through container chaining remains an obstacle. A better approach such as splitting the larger chains or restricting the chaining of medium and large containers can be beneficial, but at the cost of increasing the complexity and size of the solution space. Another approach to investigate is more offloading of reliable containers onto the core.

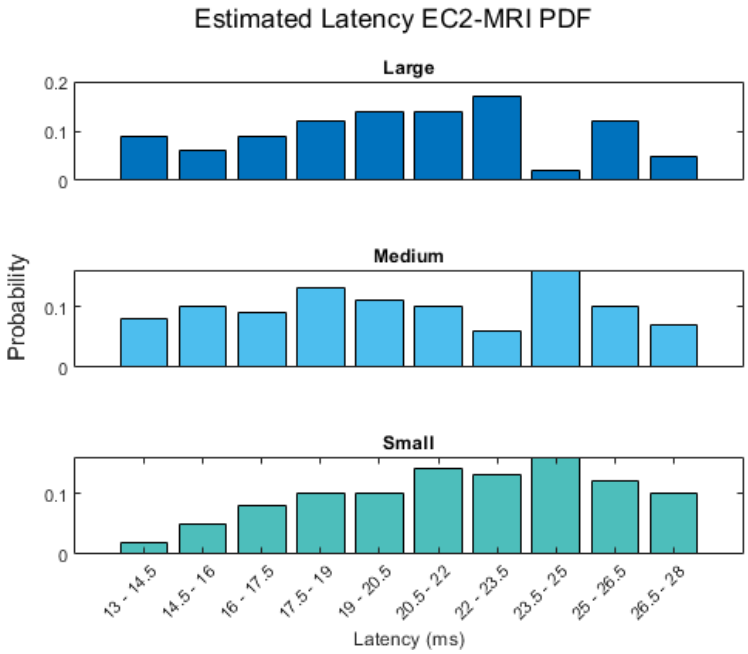

Figure 3.13: Latency experienced by the user, EC2-MRI.

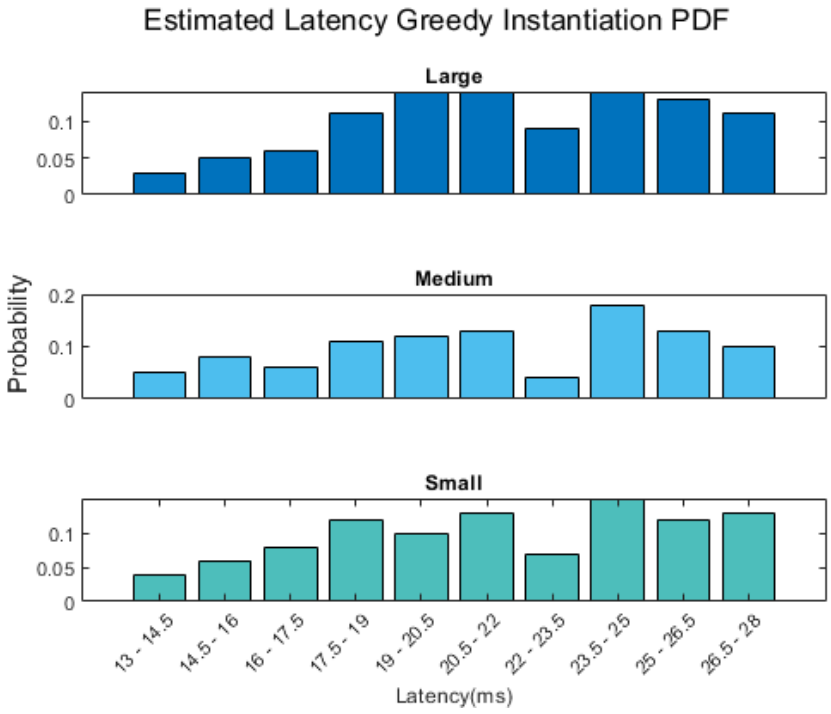

Figure 3.14: Latency under greedy Re-instantiation orchestration

3.6.3 Heuristic Analysis

While the heuristic model has a call and response aspect to it with regards to confirming placement availability before attempting to migrate or opting for re-instantiation,

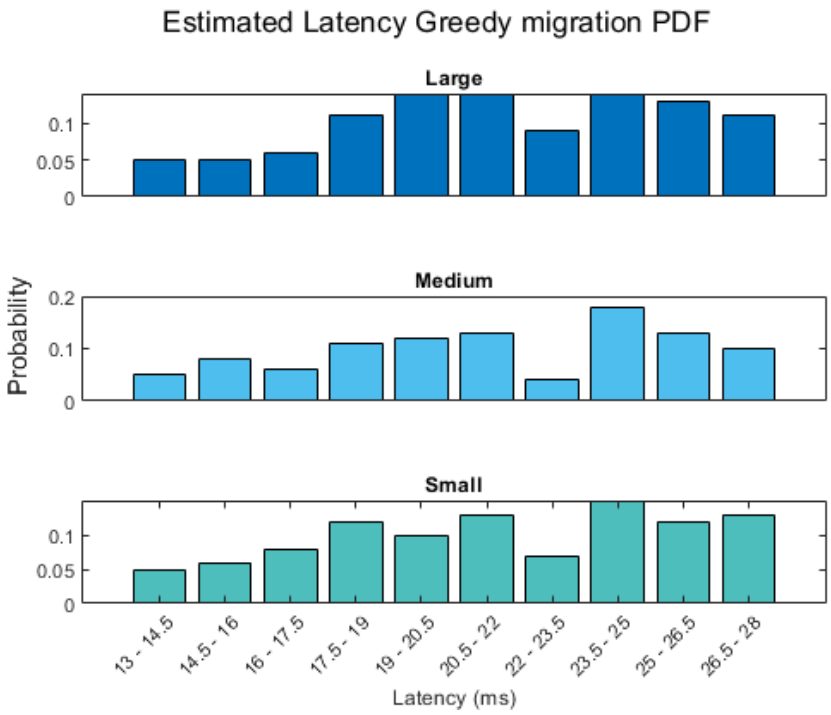

Figure 3.15: Latency under greedy Migration orchestration as per [1]

this behavior will naturally create additional overhead delay that can prolong downtime beyond preset tolerances. To investigate the amount of delay generated from this aspect of the model's functionality, Figure 3.16 shows the percentages a solution was achieved in the first two iterations for varying cluster sizes. The testing was focused on the amount of clustering for each simulation environment to ensure better stress testing through forcing longer chains to compete for a lower number of viable edge-devices capable of hosting the chains as a singular object. As shown, the system maintained a high success rate despite the varying chain lengths with little degradation when stress testing under a high rate of container chaining of 60%.

The heuristic approach segmentation is based on a unified set of clustering techniques that don't differ significantly regardless of the addressed simulation environment size. This approach is necessary to limit the complexity of the system and the variable nature of the simulation environments. The clustering techniques used showed minor improvements regardless of simulation environment size compared to the OC-MRI model. This is most visible in figures 3.9 and 3.13.

The clustering stages' accuracy has a direct impact on the EC2-MRI downtime and

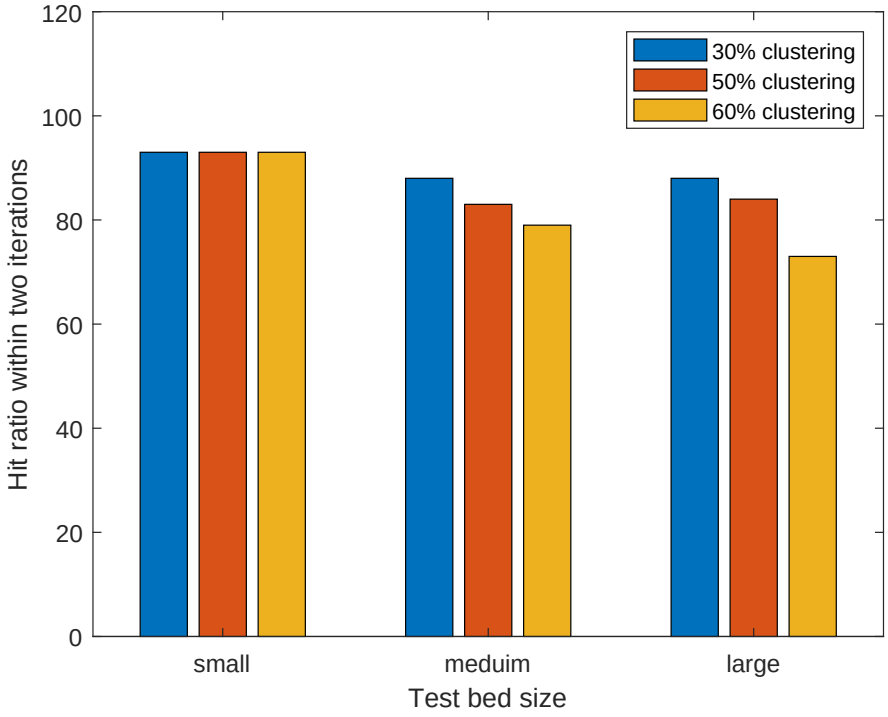

Figure 3.16: Successful placement within the first two iterations of the EC2-MRI

latencies through two means. Beginning with the clustering accuracy as shown in Table 3.6, the system's setup stage's ability to cover the solution space efficiently, creating reliable clusters, is of high importance. But another aspect that needs to be taken into account is the number of free agents generated as the overhead of core based orchestration is best avoided when aiming for lower downtimes. The table shows a high ratio of edge controlled clusters regardless of the simulation environment size, with free agents' ratio never reaching 20%.

Table 3.6: Clustering accuracy

	Clustering pre-audit	
simulation environment Size	User clusters edge control	Free agents core control
Small	92%	8%
Medium	84%	16%
Large	86%	14%

Table 3.7: Effect of auditing on error avoidance

	EC2-MRI edge fail	
simulation environment Size	No audit	Audit
Small	4	0
Medium	12	5
Large	28	7

The audit stage of the offline setup stage requires a dedicated metric when testing its efficacy. To measure its effectiveness, an isolated test run on all three simulation environments is performed with and without the auditing module. This was done to check the changes it evokes in the number of failure-generated free agents. As shown in Table 3.7, the audit stage is able to offload all errors within the small simulation environment and approximately 40% and 25% reduction during the medium and large tested, respectively, greatly lowering the cases or unplaceable container user pairs offloaded as free agents in the orchestration stage.

3.7 Conclusion

Proper placement of edge services has become increasingly critical for both network service providers (NSPs) and end-users. This work explored container orchestration in a hybrid computing environment. The various challenges hindering container edge adoption were identified and discussed. The work presented two solutions while providing detailed insights on system modeling and building blocks of a container orchestrator. The first is an integer programming optimization model, namely the OC-MRI model, that addressed the container orchestration between edge devices and core clouds. The model adhered to a number of performance and availability-aware constraints. The main target of the model was to achieve minimal downtime for fault recovery through calculating the decision variable to either re-instantiate or migrate. The model also achieved lower latencies as a secondary objective indirectly enforced through the manipulation of constraints and impact of the downtime variable. Although the proposed model minimized the downtime and provided noticeable improvements to the latencies, it remained limited by its lack of scalability and prohibitive time complexity, making real time implementation infeasible.

To address this issue, a heuristic solution was presented through the EC2-MRI

model. The algorithm consisted of two stages to maintain the overall system's ability to run in real time. The two stages allowed for the highly complex portions to be run on the core cloud where computation resources are abundant and inexpensive while the latter stage is implemented on the edge devices where real time decision making is required and computation resources are scarce. Additional metrics were used to critique the system's unique modules and their impact on the overall system's accuracy to better highlight the benefits of the EC2-MRI algorithm.

Using the current results as a starting point, we aim to convert the optimization model into a multi-objective one. This move is necessary to address the issue of the optimization cost of running the edge device as they have highly variable costs stemming from their placement and unit size. Another possible improvement is the use of the H_{mod} variable. To better utilize it and avoid conflicting magnitudes in container chains, it is best to convert it into a multiplier type modifier (values below 1 representing ranges of affinity to migrate, and values above 1 representing re-instantiation) to allow for better granularity when assigning affinity.

In terms of the heuristic model, the EC2-MRI is currently only reactive. However, a reactive approach is the natural path when aiming to lower downtime. We must first address the behavior of the system in a dynamic environment to better adjust the system. The best approach to address this issue is to start with a mobility model capable of identifying changes in the trajectories of the user clusters using the current method of subtractive clustering and making adjustments to maintain the lowest latency and highest robustness. To develop these venues of research, a heterogeneous intelligent mobility model will be necessary. Unlike the conventional mobility models currently used, we propose to use a restricted path based model that will not allow for free motion. Instead, a preset map of restricted paths representing roads and pedestrian pathways within streets or large building such as malls and stadiums, where edge computing faces the highest demands, will be developed. Lastly, we aim to use the input of the OC-MRI and EC2-MRI to train a machine-learning based model to investigate its efficacy as a real-time solution when compared to the EC2-MRI from both an accuracy and complexity point of view. This is paramount given the time-based nature of the solution and the necessary orchestration to account for failure recovery and users leaving and entering predesignated statically assigned clusters.

Chapter 4
Mobility Aware Edge Computing Segmentation Towards Localized Orchestration

4.1 Introduction

Edge Computing was introduced to allow data processing in its locality to bypass the network backbone overhead [34]. The initial design relied on dedicated servers placed near the end-users. The design was short-lived as the number of users and services employing the design grew too quickly. This is highlighted by the increase in global mobile data which is projected to grow 7-fold, with annual traffic almost hitting one zettabyte, according to Cisco's most recent annual report for the period (2018-2023) [35]. This led to the development of network virtualization and 5G networks [36]. Their development helped the edge computing paradigm adapt to expected growth by allowing the services to run seamlessly on existing edge devices and idling networking infrastructure [37]. Hence, edge computing grew from a minor portion of the entire network into a significant part of it that is no longer a homogeneous set of servers, such as those found in core clouds. Proper management of the edge computing environment became an attractive research problem [5, 38]. The concept of dynamically shifting services to serve users better emerged in edge computing orchestration. The orchestration of services in the edge environment considers the end-user's localization, services required, and the available edge devices along with their computational capabilities. Such a complex task has been rightfully delegated to the core clouds, given the edge deployment's sporadic nature which makes it challenging to take it on. However, with the increased adaptation of popular low latency services such as AR, E-health, and IoV, the idea of orchestrating off-site and engaging the network backbone becomes less feasible [39]. A new method must be introduced

to shift the orchestration process to the edge to maintain its advantages required by low latency services. While becoming more powerful and numerous, edge devices still cannot approach the core clouds' capabilities and the resources necessary to handle the orchestration process in its current monolithic format [40].

The edge resources are limited by design. Thus, increasing devices' resources to resolve this issue is not feasible. A possible approach to this problem would be the segmentation of the edge environment into more manageable pieces. While promising at first glance, this approach loses favor due to the need to properly handle several entities: services, edge devices, and end-users; each with its own set of constraints:

- Services: Share a common trait of requiring low latencies, but differ in other attributes, mainly in the amount of computational resources needed and their type; between GPU processing services in infotainment and AR apps due to heavy CPU processing such as gaming, asset tracking, and autonomous ride-sharing [41].

- Edge devices: Differ in their computational resource capabilities as well in both type and magnitude. Additionally, they hold their own unique set of constraints, such as their coverage area, energy costs, communication method, and OPEX costs.

- End-users: Between smart vehicles, unmanned drones, and pedestrians, the end users' traits vary greatly, such as in mobility from stationary to highly mobile; communication methods such as WiFi, 5G, and Bluetooth; communication format, such as V2X and D2D, and the services each requires.

When these entities are taken into account, the process of segmenting the edge space while safeguarding each of their requirements becomes harder to grasp. Thus, while segmenting the edge environment is paramount to our goal of edge-based orchestration, a simple system is not feasible. In contrast, an advanced system will be prohibitively complex based on the available resources of typical edge devices. A traditional approach of geographical grid-based segmentation would not suffice as it will significantly limit the optimal local solution. On the other hand, a highly granular and advanced clustering method can achieve low optimality differences between global and local optimal. However, it will likely falter when faced with high mobility users and spiral into a continuous

cycle of updates between subspaces due to users exiting and entering. Finally, end-users have several communication methods within the edge space that makes a geo-based approach ill-advised. Using geographical partitioning can isolate users from a local optimal placement due to geographical distance, even if their latency is better than a physically closer edge device.

Therefore, we introduce a new system to address the above challenges. The proposed segmentation system is comprised of three different modules. The first module separates high and low mobility users into two distinct layers representing pedestrian and vehicular mobility to better manage end-user mobility. The subspaces are created by clustering each mobility layer separately with different settings tailored to mobility type while allowing the newly formed layers to overlap in the coverage area. The second module virtually localizes the end-users based on their latencies instead of geo-location in a new map. Finally, the last module uses the previous two modules' inputs to resolve frequent update pitfalls and complexity limitations. It achieves this by laxly clustering the end-users while allowing more nomadic users to exist outside of the defined subspaces.

The remainder of the chapter is organized as follows. In Section 4.2, the background of edge computing and related works are discussed. Then, the design and the architecture of the segmentation scheme are explained in Section 4.3. Next, in Section 4.4, the simulation setup is summarized, and the results are evaluated. Finally, the chapter concludes in Section 4.5 with pointers to our future work.

4.2 Related Work

A majority of the existing literature focuses on orchestrating micro-service chains across geo-distributed mega-scale data centers. The dimensionality problem has recently attracted attention. Bouet *et al.* [42] offered a graph-based algorithm that takes the maximum MEC server capacity and consolidates as many users as possible at the edge into MEC clusters to maximize edge-based processing through spatial partitioning of the geographic area. Their approach considers capacity violations while allowing the cells of the same server to always be contiguous.

Guan *et al.* [43] tackled MEC region dimensionality by minimizing the number of possible handovers through clustering. They relied on a randomized algorithm dividing

a metropolitan area into disjointed clusters. The proposed system is capable of finding sub-optimal partitions that can achieve their set goals.

Tran *et al.* [44] focused on creating geographically compact clusters. They introduced a novel stochastic geo-aware partitioning heuristic algorithm that offers multiple solutions for different trade-offs between cost minimization and geo-awareness.

Lyu *et al.* [45] addressed MEC scalability problems pertaining to the massive number of devices. The author focused on resolving the issue by building a framework for offloading tasks without coordination among devices. The proposed system operated on the edge devices and involved an offloading scheme to minimize the signaling overhead.

Wang *et al.* [46] tackled user mobility by relying on the predictability of vehicles' mobility patterns and presented a 5G mid-haul design strategy that connects each Central Unit to a suitable subset of Distributed Units. The proposed system's main aim is to manage the edge computing resources required to handle peak vehicle-to-cloud V2C application load among all Central Units, showing that it can be significantly reduced.

You *et al.* [47] proposed an iterative Coverage Efficient Clustering Algorithm. The system aims to maximize coverage efficiency under delay constraints. For example, it resolves the Flying Ad-Hoc Networks (FANETs) conflict between area coverage efficiency and delay performance by alternately optimizing cluster heads, positions, and transmit powers in each iteration. The works above choose to focus on delay and cost as the objectives to minimize.

The focus of our research problem in this work is different from the problem (latency-based partitioning), the optimization objective (orchestration load and time), and constraint (solution space size) of the previous works discussed.

4.3 System Design

The system's main aim is to segment the edge environment into smaller subspaces to allow for edge-based orchestration. However, this approach can lead to lower local optimal service placements if not adequately performed due to isolating suitable candidate edge devices. It can also lead to counterproductive re-segmentation cycles if subspaces created are not stable and require frequent updates. The system breaks the process of segmenting the edge space through three modules as shown in Figure 4.1. The virtual

localization module creates a latency-based map of the total edge space. Second, the mobility segregation module separates the end-users into separate layers based on their mobility. Finally, the lax clustering module uses simple clustering techniques to create sub-spaces for each end-user type sharing all edge devices within the coverage area. Below we outline the details of each module's inner workings.

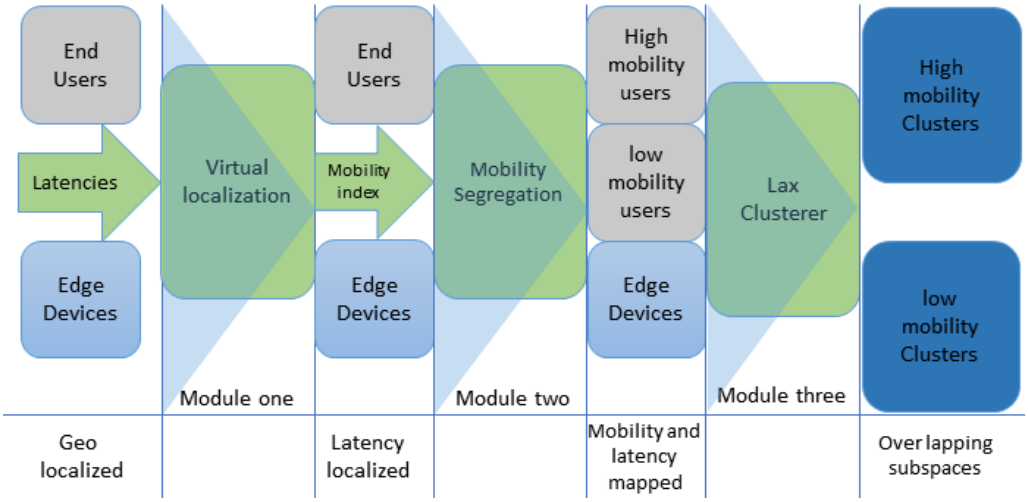

Figure 4.1: System overview

4.3.1 Virtual localization

To address the end-users heterogeneous communication nature, we make use of the virtual localization module. It sets the preliminary solution space using the latencies of the end-users and edge devices to localize them. The results are a single-layered 2D map with all users and edge devices placed based on their networking latencies concerning each other. This makes it more manageable for the latter processing modules to generate a healthy subspace capable of achieving the system's target and maintaining it.

4.3.2 Mobility-based layer creation

In this module, the localized end-user output gets fragmented into separate maps. Each user is polled for mobility and split into either low mobility representing pedestrians and stationary users or high mobility users representing vehicular and unmanned users. The results are two distinct maps of end-user overlaid upon the same edge devices map previously generated. This module's output achieves two goals. First, segregating based

on the mobility of the users further segments the edge computing environment into smaller subspaces. Second, the generated subspaces can be clustered in the following module in such a way to match their mobility better, avoiding failure conditions.

4.3.3 Lax clustering

This module uses clustering techniques to split the two maps generated by the second module into smaller subspaces. The main clustering feature is its disjoint nature, specifically the allowance of large gaps between each cluster. While the system is compatible with most clustering techniques, we choose k-means and radial clustering due to their low computational requirements when the clustering space is limited. The clustering of low mobility users relies on k-means due to their ability to isolate densely populated regions. At the same time, its shape will have little impact due to the users' aforementioned low mobility. On the other hand, radial clustering is set up with additional padding to accommodate users' mobility without allowing the subspace to deteriorate quickly. Furthermore, the clustering of low mobility users separately allows this subspace to accommodate more entities without growing too complex.

4.4 Simulation and results

To properly test our environment, a robust edge computing simulator is necessary. Several popular edge environment simulators are considered such as iFogSim [48], MyiFogSim [49], EdgeCloudSim [50], and YAFS [51]. While the other simulators are more user-friendly, including a GUI interface in iFogSim, EdgeCloudSim is chosen as the main simulating environment in this work. This choice was based on several criteria that suited our needs. The simulator is required to have:

- An easily customizable mobility and localization class to represent the end-user behavior.
- A robust generic orchestrator.
- A modular design to support robustness and easiness.

EdgeCloudSim's built-in orchestrator is based on European Telecommunications Standards Institute (ETSI) MEC orchestration method [52] capable of orchestrating, offloading, and load balancing between edge devices. The simulation is implemented on a workstation consisting of an eight-core CPU, 11GB DDR5 VRAM GPU, and 32 GB Ram.

The performance evaluation metrics used in simulation are chosen to evaluate the system performance for minimizing the solution space without impacting the optimal placements. Additionally, they are chosen while ensuring the system's robustness to user mobility, especially concerning its ability to avoid the need for frequent updates. These evaluation metrics are listed below:

- Network delay: Corresponding to the impact of the system's segmentation on the orchestration, local versus global optimal.
- Task failures due to mobility: Corresponding to the impact of limiting the orchestration space and the possibility of losing end-users during the operation post orchestration.
- Task failures due to VM capacity: Corresponding to the impact of limiting the orchestration space and the possibility of losing service during the operation post orchestration due to saturating the limited set of edge devices.
- Cluster health: Corresponding to the rate of change within the segmented subspace over the run-time.

To better represent the varied nature of services on the edge computing environment, the simulator implements four service types: AR, E-health, Gaming, and Infotainment. The services are chosen to cover the unique aspects of the popular services found in the edge. Each has a unique combination of 13 attributes, including task length, active/idle period, and delay sensitivity.

The simulation is conducted using a single simulation setup but an incremental increase in the mobile device count in increment steps of a hundred over six cycles. The test is iterated 25 times for accuracy. The results below show the simulation's aggregated outcomes for the unsegmented monolithic approach, single layer (no mobility segregation), and the proposed dual-layer clustering schemes.

Figure 4.2 highlights the importance of the virtual localization layer on the proposed scheme's performance. Without it, the network delay remains susceptible to growth with the increase in mobile devices. However, using network virtualization, both with and without mobility segregation, leads to a significant reduction in the delay end-users experience. This is attributed to promoting the latencies mapping over the geographical mapping, thus avoiding cases where heterogeneous communication methods not matching physical closeness.

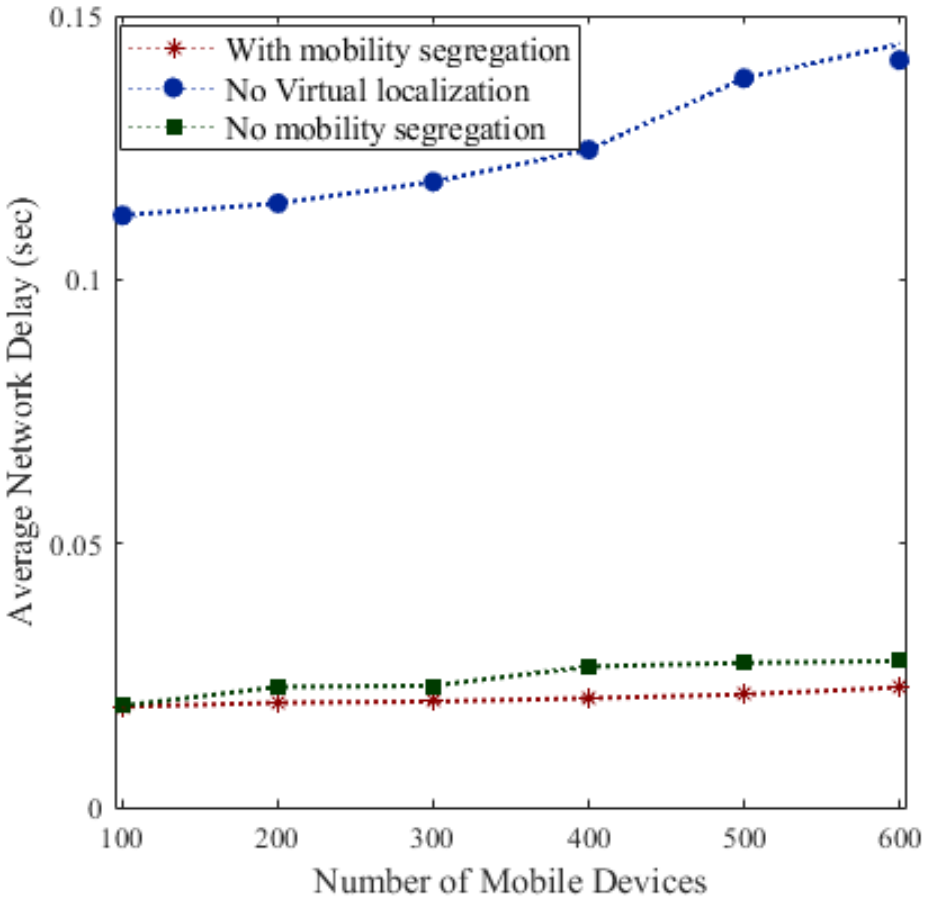

Figure 4.2: Impact of virtual localization on delay

In Figure 4.3, the scheme's performance in operating using the isolated edge resources within each subspace is illustrated. The results show that the scheme has suffered a slight setback at the lower ranges of user density. This is due to the subspace's oversaturation, causing the orchestrator to overwhelm the limited number of edge devices.

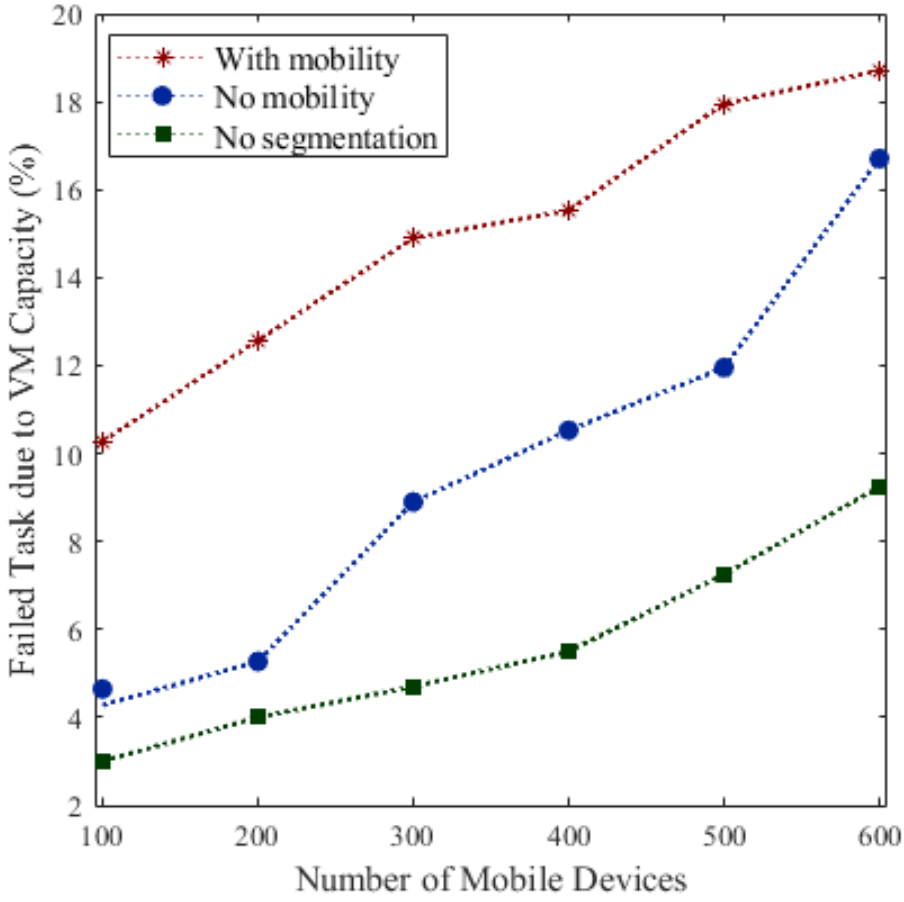

Figure 4.3: Impact of segmentation resource restrictions

The failure rate gap fluctuates within a range of approximately 7%. Upon approaching 600 users, the difference in failed tasks is much lower. We can address these issues by introducing more complex clustering techniques on the edge device layer. However, segmentation reaching such high values is rare.

The segmentation process allows for the orchestration within the edge space. However, the orchestration's limited operation while being the target can become counterproductive. A great indicator of this is the failure rate due to users exiting the subspaces. Figure 4.4 shows that the system achieves low task failure relating to the user's mobility. We can observe a growing gap caused by the mobility segregation module. It plateaued at a low failure rate up to 10% lower than the latter even when the number of users in the simulation approaches 600. This is due to the tailored treatment of mobility-aware

segmentation, such as the use of radial clusters with padding to prolong the period where a subspace maintains a user's relation at the cost of a marginal increase in the subspace members.

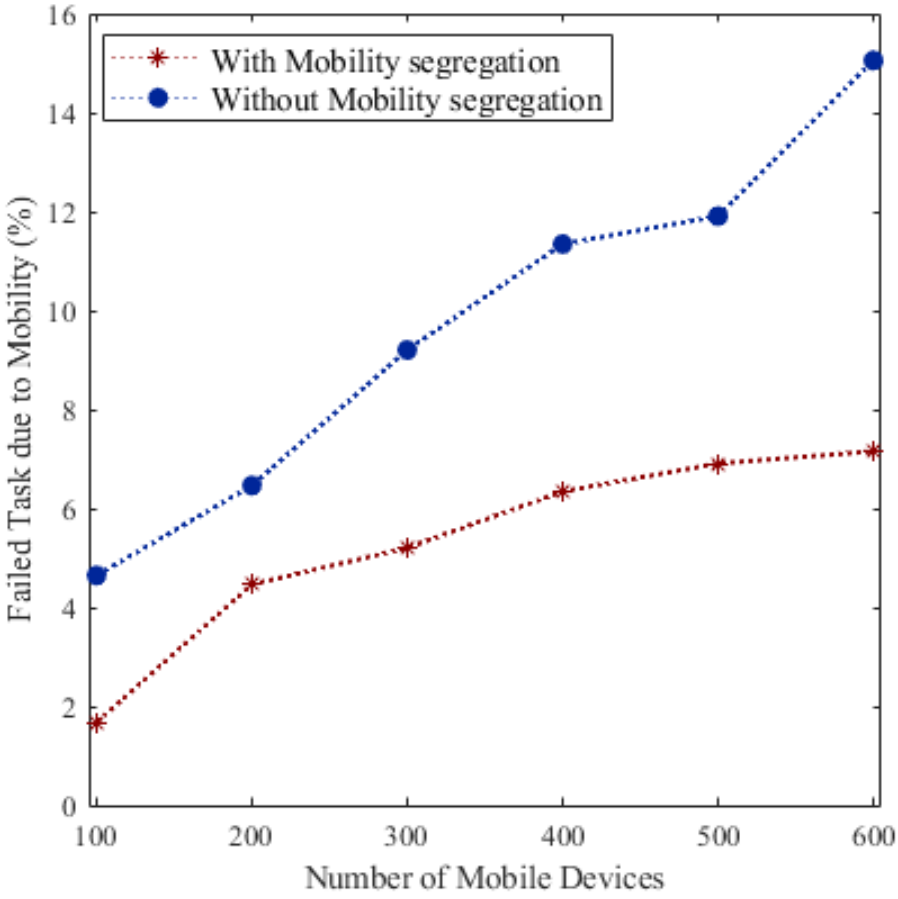

Figure 4.4: Impact of user mobility on edge space robustness

The created subspaces become an isolated orchestration problem. The duration of the subspace's viability must be considered to ensure that the subspaces created remain representative and do not require frequent updates. Figure 4.5 shows the dual-layer clustering and single-layer cluster's health degradation. The cluster health metric is calculated based on the number of clusters users lost and new users added over the total number of subspace users. This facilitates monitoring how well each subspace segmentation can maintain its original setup after several mobility cycles. The single-layer clustering (no mobility segregation) offers marginally improved cluster health compared

to its counterpart up to the 400 mobile devices simulated. However, this advantage quickly dissipates beyond that boundary with the loss of health of more than 2% compared to dual-layer clusters. This is attributed to the Lax clustering approach limiting the cluster's size based on density which remains low up to the 300 mobile device threshold, making it easy for low mobility users to exit the subspace. However, once the users number increases beyond the 300 range, the dense regions grow in number and coverage, making it more accommodating to the user's mobility.

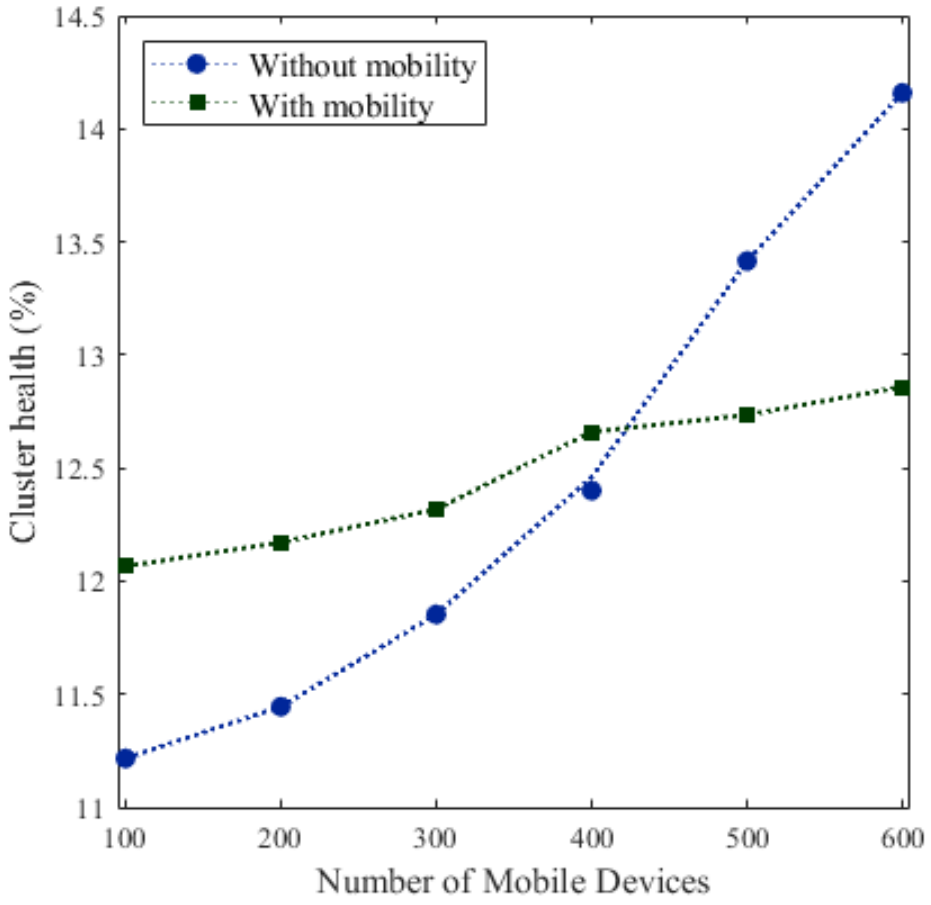

Figure 4.5: Impact of user mobility on cluster health degradation

Figure 4.6 focuses on the clustering approach and its impact on subspace robustness. With strict clustering, the system fully segments the edge space, and no user can exist outside of a cluster. While being a wanted trait, as shown in the figure, this leads to a significant degradation in the subspaces. The significance of lax orchestration is high-

lighted by its stability regardless of the increase in the number of users. Furthermore, showing the limited changes to subspaces shows that each created subspace can remain usable for much longer periods of time, thus engaging the segmentation system less often.

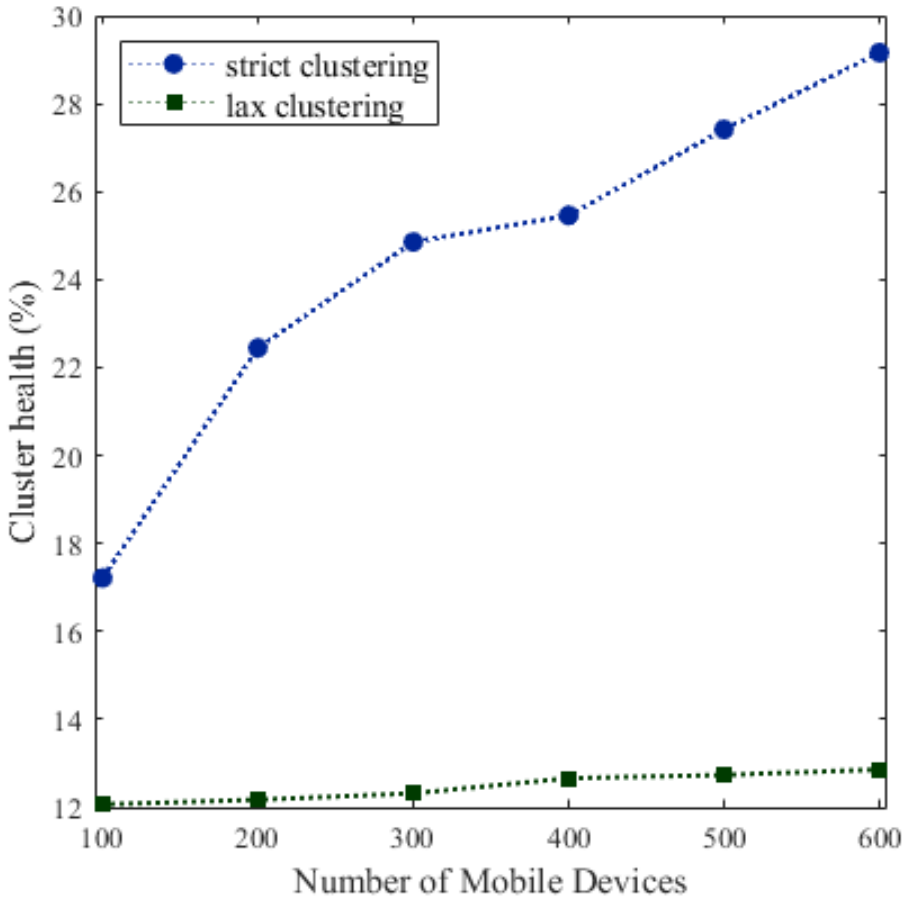

Figure 4.6: Impact of clustering method on subspace robustness

4.5 Conclusion

In this work, we proposed a system for edge computing space segmentation. The target of this system is to break down the monolithic edge environment into robust edge orchestration-friendly subspaces. To our knowledge, the approach of segregating users' mobility and virtual localization was not addressed in the edge computing current

literature. We have proposed and evaluated a three-layered system. The solution, while not optimal, remains a practical approach. For future work, our next step is to refine the clustering methods concerning mobility, specifically supporting heterogeneous users mobility types. We also aim to consider the edge device layer in our system design besides the user layer.

Chapter 5
Modular Simulation Environment Towards OTN AI-based Solutions

5.1 Introduction & Motivation

5.1.1 Motivation

The adaptation of virtualized networking architecture is currently on the rise due to multiple attractive networking and security factors, such as lower costs, high adaptability, increased robustness, reduction in latency [53], better intrusion detection[54, 55], and increased networking anonymity [56]. This push shifts the availability and placement of the networking architecture from monitored locations with technicians on hand to a more sparsely located deployment that is meant for a more automated operation approach [57]. To maintain the robustness in such setups, redundancies and other more advanced techniques are employed. Heuristic-based solutions were effective to a certain extent. Still, these solutions have reached an eventual bottleneck with the increased complexity of the networking architecture and the highly variable nature of the deployed environments. Artificial Intelligence (AI) based solutions were practical approaches in other venues due to their ability to adapt each instance based on the local deployment environment and continuous dynamic changes to maintain a high efficacy [58]. However, unlike many of their counterparts, networking-based ML solutions lack publicly available datasets. This lack can be traced back to several compounding factors ranging from network operators' fears for their users' privacy, their protection of Intellectual property and patented networking architecture, and exploitation of the datasets in finding methods to attack their internal networks [59]. This scarcity can be more apparent in contemporary networking architectures such as 5G and optical transport networks (OTNs), where new

datasets are challenging to find in a volume adequate to perform proper ML testing and debugging.

5.1.2 Available Simulators

Several recent vendor-based solutions have allowed researchers and companies to progress their work without real-life data [60, 61, 62, 63]. Their solutions are primarily subscriptions based and may entail a costly initial investment in the form of dedicated hardware, add-on libraries, or features based on the accuracy type and volume of data required. Table 5.1 outlines a few of the more popular available solutions along with their features and requirements.

Table 5.1: Simulation Environment Comparison

Simulator	Cost	Features & Requirements					
		Optical networking Builtin Mobility	5G Full Stack	Packet capture Scriptable	Local simulation Cloud service	GUI based Sumo Integration	Configurable Network Conditions Injection	Abnormal Traffic	Traffic Grooming Packet Loss	Latency	Operating System Batch operations	Unrestricted Element Count	
Opensource													
NS3 (GNS3)	Free	◑	◑	◑	●	●	●	● - ●	●	●	◑	●	◐	●	B ● ●
NS2	Free	◑	◑	◑	●	●	●	● - -	●	●	◑	●	◐	●	U ● ●
OMNeT++	Free	◑	◑	◑	◑	●	●	● - ◑	◑	◑	◑	●	◐	●	B ● ●
QualNet	Free	◑	-	●	◐	●	●	● - ◑	◑	◑	◑	●	◐	●	W ● ●
ONS	Free	◐	●	●	◐	◐	●	● - ●	◑	◑	◑	●	◑	●	W ● ●
Vendor															
NETSIM[†*]	Yearly 4500+	◑	◑	●	◑	◐	●	● ● ●	◑	-	◑	●	◑	●	W - -
ADVA	Quota based	◑	◑	●	◑	◐	◑	● ● ●	◑	◐	◑	●	◐	●	W - ◑
Optiwave Systems[†*]	Yearly 3200+	◑	◑	◐	◑	◐	◑	● ● ●	◐	◑	-	◐	●	●	W - ◑
MATLAB (Simulink)[†]	Yearly 600-2400	◑	◑	◑	◑	●	●	● - ●	●	◑	◑	-	◑	◑	B - ◑

● = Natively available; ◑ = Indirectly available; ◐ = Externally available; - = Not available;
W,U,B = windows,Ubuntu,Both; [†]has academic discounts; [*]end-user tool available

Starting with the opensource simulators, NS3 and its windows variation GNS3 boast a large scientific following. This is because the system developed a robust base that attracted many users to build their dedicated simulation tools on top of it. In addition, the ongoing support of the community makes it an increasingly attractive tool

for newcomers and veteran researchers. For 5G and OTN-based simulations, there are several NS3/GNS3-based solutions available.

- 5G-LENA is one of the NS3-based modules that can simulate 5G New Radio (NR) cellular networks. The simulator is an evolution of the famous LENA, the LTE/EPC Network Simulator with a robust uplink scheduling control as highlighted in [64]. It is currently being developed and maintained by the Mobile Networks group CTTC (Centre Tecnològic de Telecomunicaciones de Catalunya). It is a full stack simulation of the 5G NR setup that allows the user full utility and control over the simulation depth and configurations.

- Photonic WDM Network Simulator (PWNS) is one of the NS3-based modules used in the simulation of OTN-based networking. While the tools it offers are of great use, the lack of development and community updates of its base hindered its popularity as it is difficult to integrate it with more current NS3 releases.

OMNet++, similar to NS3, is a general networking simulation environment with a vital support and development community. However, unlike NS3 or GNS3, it boasts a more friendly and robust GUI-based interface, making it a more attractive option for development.

- Simu5G OMNeT++ module allows researchers to simulate and benchmark solutions on an easy-to-use framework. It offers support to optimization tools such as CPLEX and can be integrated with other modules from the INET Framework, allowing network scenarios where 4G and 5G coexist.

- The optical networking OMNeT++ module allows researchers to implement OTN-based topologies with unique structures, such as Optical switches, amplifiers, etc. In addition, the advantage of fully controlling the placement and capabilities of each component allows for better testing and integration with other solutions during testing. Moreover, its support community remains active with slight compatibility issues with the latest OMNeT++ releases.

On the other hand, for vendor-based simulators, NetSim is one of the industry's leading paid 5G NR simulation tools. It offers End-to-End simulation of 5G networks, GUI-based with drag and drop capabilities, packet animator, results in dashboard,

packet-level simulation with detailed packet trace, event trace, and NR log file generation. Additionally, it offers a fully controllable app-related user behavior and integration with mobility-based solutions such as SUMO. Unlike open source solutions, it allows for cloud-based subscription, letting researchers develop the topologies locally but run the simulation environments on the cloud, forgoing the need for advanced hardware requirements.

The vendor and open source features offer an attractive combination, but it becomes cumbersome for complete end-to-end simulations. For example, suppose the user opts for a vendor-based solution. In that case, they could be constrained by the number of users included in the license, limiting the size of their simulations, forcing multiple iterations to achieve desired results, or forced to purchase additional packages. On the other hand, if the user opts for an open-source solution, they will face setup-related issues and possible hardware-based limitations, causing similar simulation size-related challenges.

5.1.3 Contribution

To address these issues, we propose a modular-based simulation system that maximizes the user's ability to simulate 5G and OTN-based environments. The system is set up to allow multiple devices to run concurrently to achieve a more extensive encompassing simulation. Additionally, it allows the mixed use of multiple solutions, further increasing the user's throughput and adaptability. This is achieved by exploiting the use of packet files as both input and output in addition to a controlling file-triggered script to monitor and trigger the various system modules when ready.

The remainder of this chapter is organized as follows: Section 5.2 describes the simulator including all of its internal modules. Then, Section 5.3 evaluates the performance of the proposed and developed simulator using multiple testing methods. Moreover, it discusses the achieved results. Finally, Section 5.4 concludes the chapter.

5.2 Simulator Description

The system comprises four independent components to maintain its modular nature, with the outputted files being the only interaction between them, as shown in Figure 5.1. This allows the users to capitalize on the systems' two main benefits. First,

it allows for both single and distributed operations by relying on the script to direct and synchronize the number of devices used using file creation as triggers. Second, the system allows multiple simulation methods for 5G and OTN. In what follows, a brief description of each of the four modules is provided.

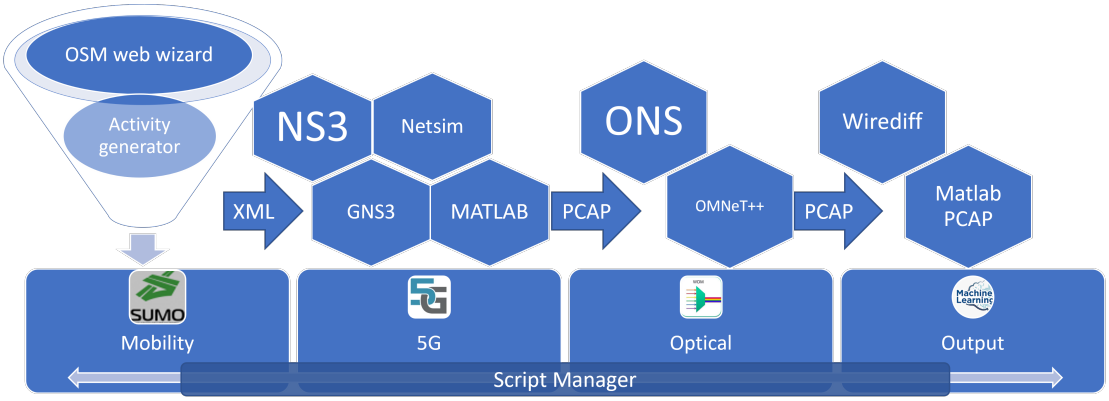

Figure 5.1: System Design.

5.2.1 Mobility module

This module is responsible for mimicking the real-life nature of the mobility used in the simulator. This is achieved by using SUMO and OSM wizard. The user begins by isolating the map section to be simulated using the OSM web interface and allocating the number of vehicles and pedestrians. Once generated, the user can further enhance the quality of the mobility simulation by editing the city properties XML files to replace generic values and assign more realistic ones. This creates sumo-compatible demands for a synthetic population by adjusting population-related aspects.

- Population:
  - Inhabitants and demographic division
  - Households distribution
  - Employment patterns
- Location-based:

  - Bus lines and stops.
  - Schools, workplaces, Malls.

- Time-based:
  - Work hours
  - Education hours
  - Opening/ losing times for specific locations like Malls, stadiums etc.

This can also help generate abnormal test events as the user can create massive congestion or everyday public events. This is important for proper testing of ML-based solutions to ensure the training and testing include aberrant events improving the quality of the resulting models.

5.2.2 5G module

This module attaches 5G-based traffic demands to the mobility captured earlier. Unlike the mobility generation stage, the 5G module can leverage pipelining techniques to increase the simulation output volume. This is done in multiple ways based on the 5G simulator used. The most common open-source application for this is NS3 or its windows-based variation GNS3, and the vendor-based Netsim application. The script first prompts the user to check for the mode of operation, either singular or distributed, and choose the available applications. Based on that selection, if the language is fully scriptable (as outlined in Table 5.1), the script can invoke the application using the sumo file as bases or, in the case of Netsim, can prepare and guide the user through the GUI-based steps necessary to get the simulation started. In either mode, the PCAP file generation will be capped using the XML sumo file outlining the number of communication elements in each setup. This will trigger subsequent steps once that number of PCAP files is achieved or a timeout is reached.

Once the simulation is finalized, the PCAP files are collected and renamed using batch renaming software to conform to each app's varying naming convention in cases where the file naming is not fully customizable. The output is then saved in the master simulation folder.

5.2.3 OTN module

Currently, the OTN module has been tested only on OMNeT++. The script feeds the PCAP files using some of the software's built-in functions. The OTN network is built using disjointed pairs to allow various path variations to occur concurrently. The OMNeT++ simulator offers a fully customizable OTN structure, including editable components such as optical amplifiers, switches, lines, receivers, etc. The traffic grooming is generic but can be customized using community-based libraries accessible to the users. Once the resulting PCAP file reaches the final destination node, the script captures the file edit and triggers the OTN output.

Each PCAP file stream is truncated to isolate only the initial input and final output. It is then renamed based on the User ID initiating node and destination nodes. This information is used later on and extracted by the script to generate the networking latency metrics.

5.2.4 Output processing

After the PCAP files begin populating from each simulation environment, the script can trigger idling local resources to start processing and extract relevant metrics such as latencies, packet loss, error rates, etc. This is done using two methods: MATLAB for scripting only, and Wirediff for more graphic and open-source options. Once the data has been collected, it is saved as either a MAT File or a TXT file based on the user selection.

5.3 Performance evaluation

5.3.1 Objective

To ensure the system's usability for ML purposes, we need to ensure that its overall capabilities match or exceed those of its individual components working separately. To best test, two main attributes are identified as the focus of evaluating the quality of the data collected as a whole and the number of differences between each variation. This is done to ensure that the data the variations output is cross-compatible, a crucial attribute to ensure the correctness of any ML-based solutions built based on the dataset.

5.3.2 Metrics

The metrics used to measure the effectiveness of the simulator are chosen to offer a comprehensive view of the system's functionality. The three metrics are:

1. The similarity of the data outputs of the different 5G module variations in the form of an encompassing conformity score.

2. The simulation time required for the components tested for both monitored and automated portions to highlight the human factor required.

3. The amount of data collected during those simulations in the form of users per simulation session of comparable length.

5.3.3 Testing methods

To properly test the system capabilities given the intended accessibility, the testing is done on three desktop devices with identical capabilities limited to average specifications. The testing is done for both singular and distributed setups. The variations tested are limited to the windows-based solutions with GNS3, and Netsim used for the 5G traffic simulation, while the OTN is handled using OMNeT++. The following tests are conducted:

- Singular mode with a static sumo environment running independently on multiple 5G-only environments running a unified single application type on all users.

- Singular mode with a static sumo environment running independently on multiple variations of the entire system.

- Distributed mode with a static sumo environment running collaboratively on both the cloud Netsim and NS3 variations.

- Distributed mode with multiple sumo environments running collaboratively on the cloud Netsim and NS3 variations.

- Distributed mode with multiple sumo environments running on NS3, running collaboratively.

The above combination of tests provide full coverage of the most common use scenarios of the system with a focus on the volume of data produced, its conformity with variations, and the quality it provides for use in ML discussed below.

5.3.4 Results

To showcase the system viability, we first test the output of the 5G generation module variations to ensure their output can exist within the same data stream for ML usage without creating any biases or related issues. Figure 5.2 shows the output conformity (calculated using equation (5.1)) of simulating Video streaming, VOIP, and File transfer services on all users in the following environments: NS3, OMNeT++, and NetSim. The results show high conformity of the data outputted across all three variations compared to Netsim with a slight reduction in VOIP traffic. These high levels are achieved due to the highly configurable nature of the application deployment in all three applications, with NS3 and OMNeT++ offering the most control.

$$Conformity = 6(Latency_{req}) + 4(Demand_{dur}) + 2(Demand_{freq}) + 3(Packet_{AvgSize}) \tag{5.1}$$

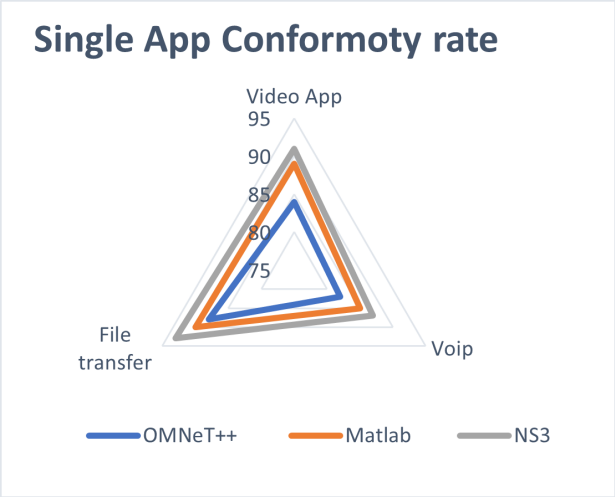

Figure 5.2: Single app conformity compared to Netsim.

However, manually adjusting the application deployment is neither scalable nor compatible with the system's goal of low maintenance. Therefore, to test the system's automatic attachment of applications to users based on the demographic data supplied

in SUMO, we test the system mentioned above. Figure 5.3 shows the system's conformity after allowing them to use built-in functions applicable to manage the application distribution among the users. Results show slightly lower conformity but remain within an acceptable range for ML purposes. The increased variation can be positive, especially when building resilient ML-based systems.

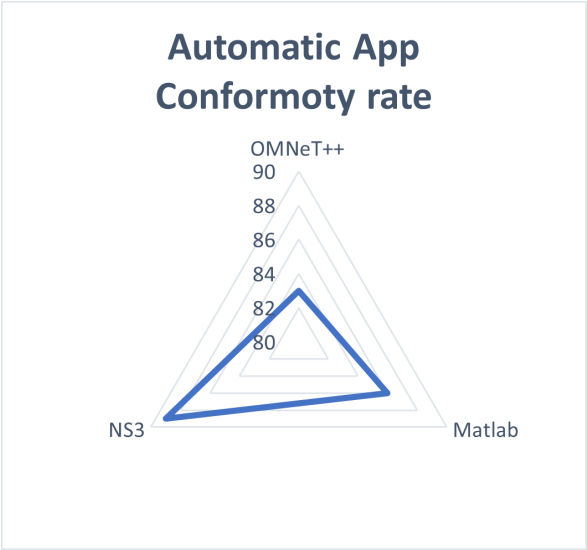

Figure 5.3: Heterogeneous app conformity compared to Netsim.

After ensuring that multiple system variations are viable, we shift our focus towards the quantity of data generated. Two main cases are in consideration:

1. Creating multiple runs on a single geographic area best suited for testing related to 5G and edge computing.

2. Creating a massive simulation over multiple geographic areas.

Figures 5.4 and 5.5 compare the aggregate simulation runtimes, including the duration of required user interaction for hybrid and pure open-source distributed-based solutions.

Table 5.2: 5G Simulator Output Volume

	Number of Users
OMNeT++	+900
Matlab	+700
NS3	+900
Netsim	500

The results illustrated in Figure 5.4 show that the use of Netsim required the most monitored setup time. At the same time, it offloaded the need for local hardware resources but required more user intervention in the setup stage due to its method of sumo mobility extraction. On the other hand, Figure 5.5 shows the NS3-based solution, and its highly scriptable nature allowed for better-automated simulation. Another aspect to consider, shown in Table 5.2, is the main shortcoming of using Netsim due to licensing-related limited number of simulated users. While it is considered a hindrance to the local solution and offers a slight increase in the number of simulated users, it requires more time and reservation of the resources.

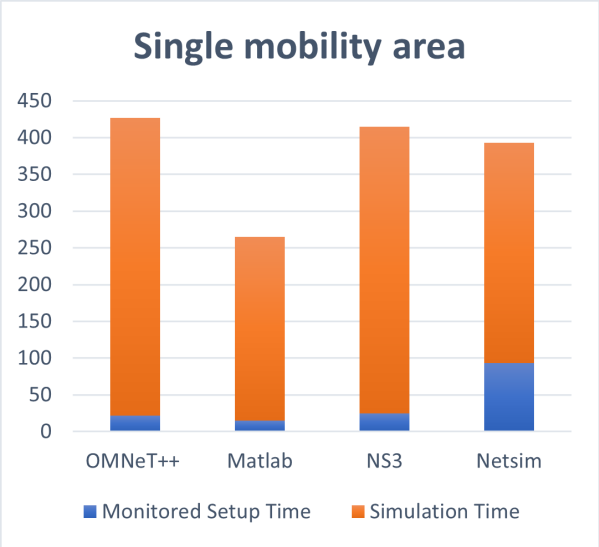

Figure 5.4: Simulating iteration of a single limited size environment.

5.4 Conclusion

In this chapter, we presented a viable simulation system that uses a combination of state-of-the-art environments to address the ongoing issue bogging down the development of AI-based systems in next-generation networking environments, especially in 5G and optical networking. The system considered combinations of paid and open-source solutions to work seamlessly to achieve the highest diversity and volume of data possible. The system testing results show considerable improvements over traditional approaches with minimal increase in the requirements or the need for dedicated hardware. The system can be improved further by developing the pipelining apparatus controlled by the

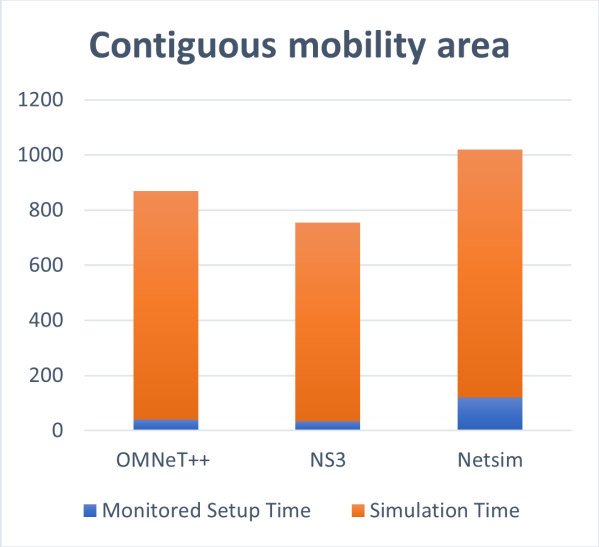

Figure 5.5: Simulating a single large environment.

script to allow for better handling of the live feed of PCAP files while the simulation's earlier stages are ongoing to reduce the needed runtimes further. In addition, packaging the system using OS or VM images reduces the necessary setup and know-how to create customized datasets.

Chapter 6
Throughput Latency Targeted RL Spectrum Allocation In Heterogeneous OTN

6.1 Introduction

The current trend in user services have become more complex and stringent in nature, with 5G setting the trend with its own set of core services, including enhanced mobile broadband (eMBB), ultra-reliable low latency communication (URLLC), and massive machine-type communications (mMTC) [65]. Some of these services require high bandwidth and low latencies, often simultaneously. Traditional networking archetypes are not able to fulfill such requests while acting as the main backbone[66]. Thus, the adoption of optical networking is more critical. However, while optical networking can handle such large volumes of data, the physical nature of using light as a medium presents unique challenges. The rigid physical nature of using light as a medium limits the system's ability to continuously optimize its network payload, especially with a projected larger and highly varied traffic throughput than traditional networks have ever experienced. This mismatch creates a need for multiplexing the low-rate traffic demands into optical wavelength channels. However, this process involves using electronic switching equipment, which is counter-intuitive to optical networking, offsetting the benefit of continuous optical transport.

To address this, the concept of spectrum allocation was developed, which entails changes to network design and resource allocation algorithms to enable better network bandwidth through lower electronic switching. For a period of time, this was sufficient and allowed optical networks to easily surpass traditional networking. However, all the algorithms and solutions that were investigated reached a bottleneck in terms of possible improvements. This is largely due to the rigid spectrum granularity employed by the prevalent methods in the archaic fixed grid optical transport networks (OTNs) [67]. In

recent years, there has been an improvement in spectrum allocation, with the ability to allocate it flexibly or "slice it off" based on the demands in elastic optical networks (EONs) [68]. The migration to EONs from fixed-grid has proven to be effective, particularly in meeting the demands of 5G-based services. The main methodology used was to reduce the need for electrical re-multiplexing by assigning optical slots in wavelength division multiplexing (WDM) for grouped stable demands through a process called routing and wavelength assignment (RWA) [69]. Initially, the focus was on maximizing the survivability of the networked traffic in the early optical networking architectures. In recent years, it has developed into routing and spectrum assignment (RSA), with a shift in focus towards lowering the need for electronic switching to maximize the benefits of modern optical networking architecture [70]. However, with high transmission rates from 400 Gbit/s to 1 Tbit/s, the variably segmented spectrum slots range from 37.5 GHz to 400 GHz. A disorganized placement of optical demands within a lightpath can create numerous variably sized placed empty slots that are sporadically placed. This occurs when demands are eventually extracted while traversing the various optical nodes. This phenomenon is referred to as spectral fragmentation. It can be challenging to reuse the freed bandwidth resources due to the increased blocking probability.

To better illustrate this problem, particularly in a futuristic environment, Figure 6.1 depicts a use case were traditional spectrum allocation (SA) fails to correctly allocate the URLLC traffic generated during limited events, such as a metaverse virtual reality (VR) concert. Unlike traditional broadcasted events, such as televised sports events that mainly involve latency-tolerant downstream traffic, the VR concert involves tens of thousands of virtual attendees who view the main event while interacting with each other via their VR avatars. This requires both downstream and upstream traffic that are latency-sensitive, with sessions that can last from a few minutes to 3-4 hours [71]. Such events will become more frequent in the future. The figure highlights how ignoring the short-lived nature of such demands results in more fragmentation, while the bottom portion shows a better placement that results in fewer fragments.

Consequently, fragmentation causes future demand allocation to get delayed until the next time slot, hoping to land on large enough contiguous slots or trigger a reallocation to create sufficient slots. Both options are either costly or time-intensive, making them counter-intuitive for the purposes of OTN. To resolve this issue, the current state-

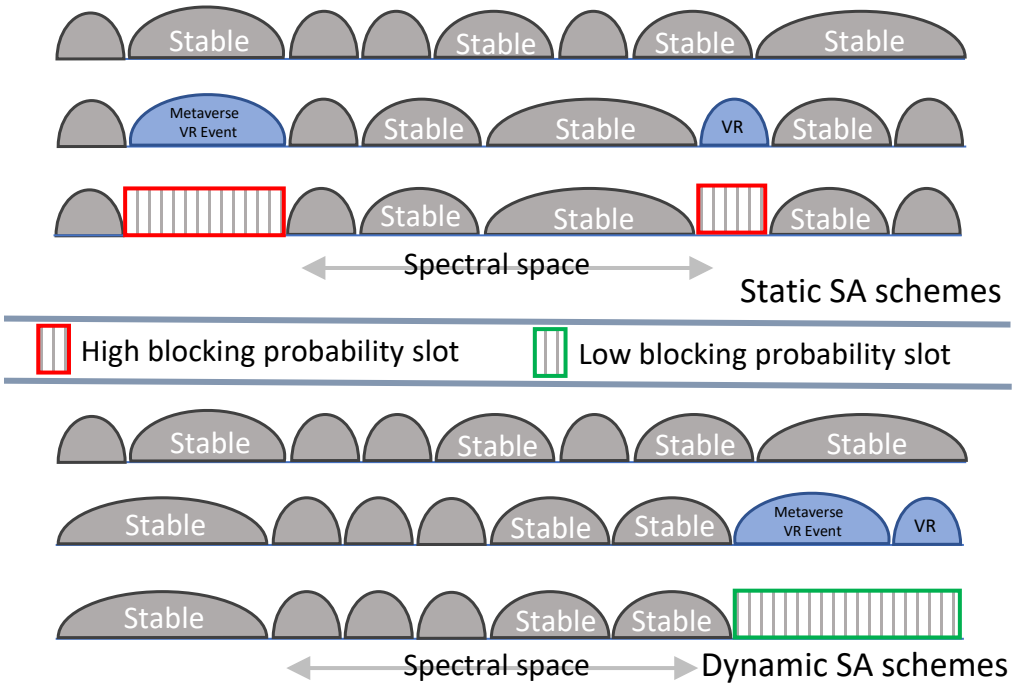

Figure 6.1: Dynamic demands impact on fragmentation

of-the-art research proposes resource allocation schemes to reduce fragmentation. These proposed approaches are mainly designed at the electrical and optical layers, which can be categorized into "Intuitive", "Straightforward", and "Conventional" spectrum allocation approaches. The latter can be further categorized into "First", "Random", "Last", and "First-Last" Fits. In addition, slightly more advanced methods rely on historical usage patterns, such as the least and most used slots. However, even with significant improvements when using reactive and proactive methods, such approaches may quickly result in a similar bottleneck, as noted by Chatterjee *et al.* [72]. Due to the abundance of available spectrum in the past, the main shortcomings of the approaches that garnered attention were focused solely on lowering energy demands [73] and reducing the required optical infrastructure [74]. However, even with the major advances in WDM-based optical networks, the wavelength continuity constraint remains challenging in current networking demands. This constraint requires the same wavelength to be used on all hops in the end-to-end path of each connection and limits our ability to adjust or reroute demands mid-path without incurring prohibitive costs in both energy and time. Additionally, the highly virtualized and customizable 5G networking infrastructure creates a different impact constraint stemming from the rapid changes the network can experience based on

the time of day and other factors. Traditional SA schemes do not adequately account for this high dynamicity.

In this work, we design a machine learning (ML) based spectrum allocation algorithm capable of capturing the varying dynamicity of each node and making live adjustments to the optical path spectrum allocation process to maintain low fragmentation and latency typical of EONs, namely deep reinforcement learning (RL) this specific type of ML was chosen because it suits the solution method given our node specific approach and our need to tune our spectral allocation approach to the unique set of demands each node experiences at specific times of the day achieving a better fit compared to globally developed optimization techniques or other types of ML [75]. The main contributions of this work are:

- Minimizing bandwidth fragmentation by designing a fully observable environment-based Q-learning RL model.
- Operating in a non-episodic continuous environment iterating actions from previous cycles till the model output plateaus.
- Introducing a dynamic search window to isolate the reward and punishment mechanism to increase the accuracy of the critic's output and avoid jitters.
- Reducing the scheme actor complexity while maintaining low overhead by implementing bidirectional spectrum allocation that focuses on pushing the edges while clearing the central portion of the spectrum.
- Making node-based offloading decisions to traditional networking infrastructure, relying on the individual demand's latency constraints.

The remainder of this work is organized as follows. In Section 6.2, the motivation and problem definition is outlined. In Section 6.3, the current related works are discussed for the fixed grid, EONs, and their hybrids. Section 6.4 outlines the model operation along with the proposed spectral slicing process. Section 6.5 discusses the simulation environment and the various metrics used to benchmark along several baseline algorithms. Finally, Section 6.6 concludes with a brief summary of the achieved results and outlines plans for future improvements.

6.2 Motivation and Problem definition

In the early stages of optical networking, the advantages of lower power consumption and higher bandwidth were the main attracting features for industry and researchers. As a result, most of the research focused on attributes such as survivability/resilience and energy consumption. This direction in research was reasonable at the time, given the type of traffic being serviced. However, this changed with the rise of 5G applications with the new URLLC classification that requires ultra-low latencies while still demanding heavy bandwidth. Such services include augmented reality (AR)-assisted surgery, live sports events streaming, and online gaming among others [76]. The current trend expects those requirements to increase under 6G networks and beyond. Furthermore, network traffic and user connectivity are expected to grow as the number of mobile 5G subscriptions worldwide is expected to reach 4.8 Billion by 2026, posing further challenges for these networks [77]. These trends have shifted the focus from energy preservation to latency and throughput.

One of the main challenges of optical networking compared to traditional electrical networking is the handling of data streams. In contrast, the best solution currently favored by the majority in optical networks is spectrum allocation. These methods mainly rely on intelligently organizing traffic demands before converting them to optical format to reduce or eliminate the need to handle this task before arriving at their final destination.

Previous research on spectrum allocation relied on advanced mathematical formulations in the form of optimization models and heuristic algorithms to handle the demand organization task. However, while creating benefits, these approaches are costly from both time and expenditure points of view. Additionally, the deployment environment must significantly mimic the development environment to ensure proper operation [78]. With the highly dynamic nature of next-generation networks (NGNs), 5G included, such an assumption is not reasonable. To address this, machine learning (ML) solutions were sought in the field of OTN spectrum allocation with trend-setting positive results for energy purposes. This opens a new path to address the issue of servicing the new demand paradigms, such as URLCC, within increasingly dynamic environments typical in the upcoming NGNs as illustrated in Fig. 6.2 showing the impact of treating all demands

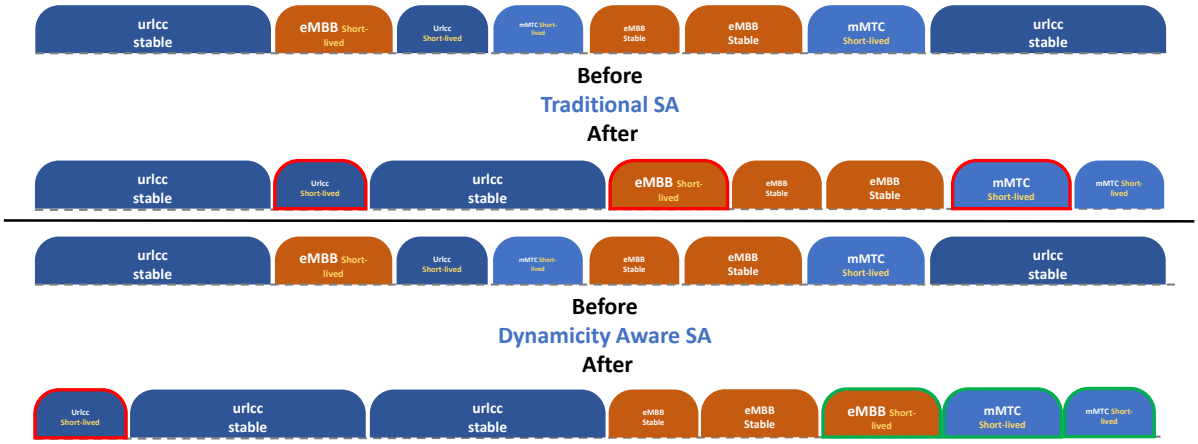

Figure 6.2: Traditional vs. Dynamicity Aware SA

equally from a duration point of view where short-lived demands can create a large number of fragments when placed along long-lived demands instead of trying to distance or isolate them from each other.

6.3 Literature Review

To best capture the current academic research direction and the related issues, the section has been divided based on the targeted problem being addressed, mainly focusing either on the spectrum fragmentation in EONs, traditional OTN setups, or combinations of them. The subsequent section focuses on the shift of focus toward the dynamicity of serviced demands and its impact on the aforementioned spectrum allocation solutions. It also proposes approaches to mitigate its impact.

6.3.1 Spectrum Fragmentation

To best capture the current academic research direction and the related issues, the section has been divided based on the targeted problem being addressed, mainly focusing either on the spectrum fragmentation in EONs, traditional OTN setups, or combinations of them. The subsequent section focuses on the shift of focus toward the dynamicity of serviced demands and its impact on the aforementioned spectrum allocation solutions. It also proposes approaches to mitigate its impact.

6.3.2 Spectrum Fragmentation

Given the rigid nature of light transport, proper planning in the form of spectrum organization ensures better usage of each lightpath by allowing the highest usage ratio in the initial node and lowest blocking probability in subsequent intermediary nodes. Even before the widespread of EONs, researchers foresaw this drawback and created different approaches for each unique case.

Yu *et al.* [79] took into account the gradual shift towards elastic optical networking and the increased strain it places on SA solutions, which increases the blocking probability. The authors tackled the spectrum fragmentation problem by creating a unified system aiming solely on lowering the bandwidth blocking ratio (BBR) compared to the state-of-the-art RSA algorithm. The proposed solution showed improved performance in more highly connected networks, which they expect to be the future of OTN topologies.

Papanikolaou *et al.* [80] formulated a joint multi-layer planning problem and proposed two distinct integer linear programming (ILP) formulations to solve it, achieving significant cost and power savings.

Zhao *et al.* [81] proposed a novel model based on a mutual backup that can improve the control plane's survivability in a software defined elastic optical network (SwD-EON), providing survivability to unicast and anycast traffic demands. They formulated an ILP to find the optimal solution with additional spectrum resources required when survivable multipath routing is used.

Weiqi *et al.* [82] introduced a heuristic algorithm for spectrum allocation, referred to as "heuristic," which considers path delay, optical energy use, and electrical energy consumption. Their devised approach effectively reduces the average path delay without affecting energy efficiency. This was achieved by utilizing the 'Frequency First' strategy, leveraging the distribution of IP links (IPLs) to lower the average path delay without adversely affecting energy performance.

Kaur et al. [83] pursued a more unique heuristic approach based on a wind-driven technique to optimize spectrum allocation in optical networks in terms of hyper-volume and set coverage indicators boasting consistent improvements under varying load conditions.

Finally, Lee *et al.* [84] proposed a new bio-inspired spectrum allocation algorithm, boasting a short convergence time when performing lightpath conversion in the IP-over-

WDM networks. Their solution's main advantage relies on variations to ossified local solutions, improving the possibility of finding a better global solution and reducing up to 20% of energy consumption and computation time compared to other heuristic algorithms and mutation-based conventional genetic algorithms.

6.3.3 Dynamicity

Next-generation networking has brought not only an increase in traffic volume, but also a variety of traffic types. Consequently, this has led to a significant increase in the dynamicity of serviced demands in optical networks. To address this, traditional sporadic offline spectrum allocation (SA) approaches have shifted towards live proactive solutions. Researchers have proposed various ML and big data analytics techniques to overcome the challenges that cannot be handled using traditional approaches, making them highly beneficial in optical communications and networking.

Manias *et al.* [85] introduced a solution that tackles dynamicity by intelligently allocating optical resources to provide robustness in the network to handle any fluctuation in demand with minimal over-provisioning using optimization modeling as well as heuristic solutions that adjust their configuration based on the traffic and optical paths they manage. Khan *et al.* [86] proposed that ML and big data analytics can provide better outcomes for optical networks, especially in the face of their increasing dynamicity and software-defined nature. Yang *et al.* [87] proposed a dynamic unsupervised fuzzy clustering scheme for achieving higher accuracy, while Zhao et al. [88] created self-optimizing optical networks. On the other hand, Musumeci *et al.* [89] gave a broader view of the impact of using ML on EONs, along with a few potential ML-based solutions. Yan *et al.* [90] tackled resource management using reinforcement learning. Similarly, Chen *et al.* [91] used Deep Reinforcement Learning for a total EON solution that tackled routing, modulation, and the spectrum assignment (RMSA) processes simultaneously. Yang *et al.* [87] proposed a mechanism for achieving zero-touch operation in optical network architecture. Proietti, *et al.* [92] introduced an ML-based quality of transmission estimation scheme for lightpath provisioning with intra-domain and inter-domain traffic.

6.3.4 Limitations of Previous Works

The previous research has provided valuable insights into various aspects of optical network management. However, there are still some gaps to be filled. Although the existing works have made significant contributions in reducing energy consumption, improving throughput, and minimizing fragmentation, they have largely overlooked the coexistence of traditional networks and their compatibility with optical ones. Moreover, while AI-based solutions have shown promise in improving demand routing and organization within the spectrum, there is still room for further improvement. Therefore, in this study, we aim to focus on the organization technique employed by our proposed solution to offer a more tailored and customized approach per region/node, addressing the aforementioned gaps.

6.4 Throughput Latency First Reinforcement Learning (TLFRL) SA Model

To achieve our objective of improving throughput by optimizing spectrum utilization, we adopt a node-specific approach. Each node is individually configured to enhance the subsequent nodes' ability to handle additional demands. Reinforcement learning is used due to its ability to leverage the current network status and configuration to optimize nodal models, leading to more resilient models that can dynamically adapt to the changing demands over time [93]. The next subsections detail the system's two primary modules, the actor and critic, along with the environment-specific components developed alongside them to further improve their accuracy.

6.4.1 Environment Setup for SA in Optical Networks

The system employs a solution window of a specific length to identify the center and classify the direction of spectrum allocation. This is then followed by creating a table of all the distances between the nodes used mainly by the critic that estimates the value function in the RL environment. The slot size for the flex grid is set to half that of the smallest demand to ensure sufficient granularity without increasing the complexity unnecessarily. Figure 6.3 illustrates the key elements in setting up the environment for

optimal spectrum allocation in optical networks showing the solution space split and enforcement of grooming direction based on the original spectral position of the groomed demand.

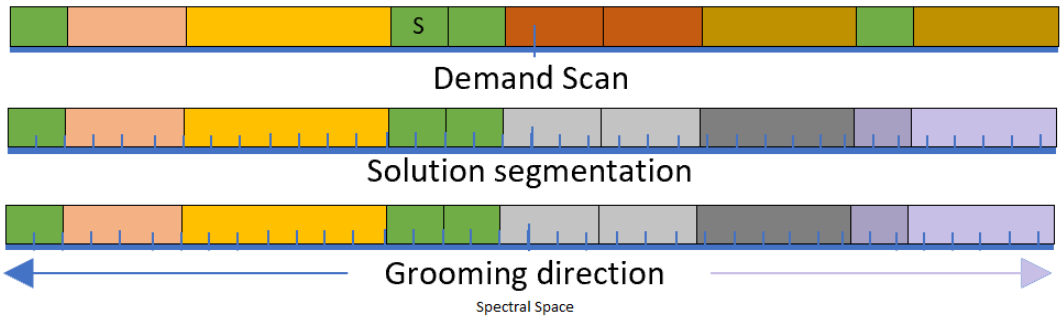

Figure 6.3: Environment Illustration

6.4.2 The Actor in the RL-based SA Proposed Solution

The actor's actions are determined by the critic's feedback for each demand after the first iteration. Figure 6.4 illustrates the cluster detection stage and a spectrum allocation example where the demand follows the allocation direction-based on its position relative to the center of the solution window.

The clustering cycle begins by searching for immutable demands i.e. those that are not under the control of the current allocating node (already placed/groomed). Once found, two anchor clusters are designated based on the longest chain of contiguous immutable demands. The anchor cluster is then used to allocate spectrum slots based on the magnitude of the penalty for each demand.

The offloading action, illustrated i Figure 6.5, is determined by the demand's latency tolerance and the cumulative score from the previous and current rounds. When the current utilization level of the total solution space surpasses a dynamic threshold, it is offloaded.

6.4.3 The Critic in the RL-based SA Proposed Solution

The critic is an integral component of the reinforcement learning system, which takes the results of the actor on the environment and outputs its own quantitative analyses to guide future cycles of the model behavior. Similar to the actor, which focuses

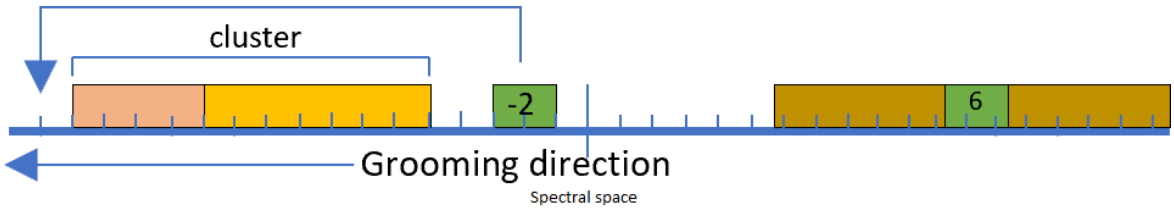

Figure 6.4: Spectral allocation action

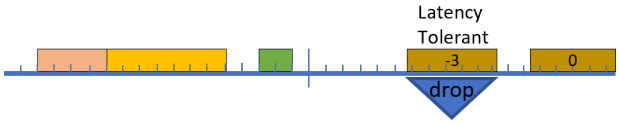

Figure 6.5: Offloading action

on demands and their placement, the critic output is directed towards each demand's spectrum to scale the reward/penalty allotted based on the distance between the allocating node and the critiquing node. This is shown in Figure 6.7 where n is the spectral allocating node, and i is the critiquing node. another aspect that had to be taken into account is the Average demand size each node experiences and how to use it to more accurately calculate what would be considered a fragment Fig. [6.6] visualizes the method we chose to calculate split our spectrum and identify fragments. Equation 6.1 eliminates the inherent biases toward offloading for all demands approaching their final destination node, causing over usage of the offloading function and negatively impacting earlier nodes' scores.

The RL actor's behavior outlined earlier is isolated in nature to its root node; however, its impact cascades impact to all subsequent nodes making it imperative to find a method to capture to properly guide the actor's future iterations. The critic behavior outlined in algorithm 4 shows the use of the scaling factor shown in Fig 6.7 to penalize the fragment-causing demands based on the overall distance traversed with induced fragments.

$$scaledscore = \frac{d_i * score_i^{demand_n}}{\sum_n^i d} \tag{6.1}$$

The search window is dynamically sized based on the groomed demand wavelength width as shown in Equation 6.2, where $TC_{searchwindow}$ is the window facing the center, and $AC_{searchwindow}$ is the window facing away from the center. This is done to ensure

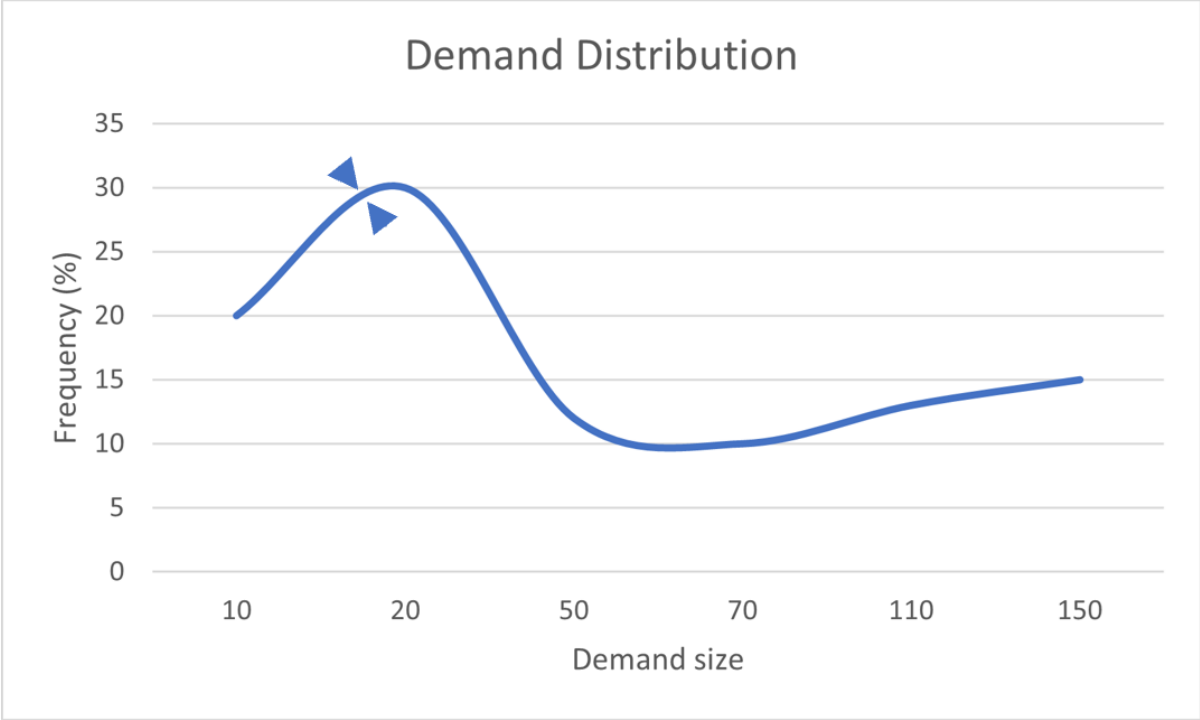

Figure 6.6: Demand distribution illustration example

Algorithm 4 TLFRL Critic

Require: $i_{ImmutableDemands} \geq 0$;
$A_{ActionableDemands} \geq 0$

while $searchposition \neq seachwindow$ **do**
  if $searchslot_n \neq utilized$ **then**
    $scaledscore_{A_n} - -$
  end if
  if $S_{d^i_{width}} > Xoffloaded$ **then**
    $scaledscore_{A_n} - -$
  end if
end while

that the areas closer to the edge are more impacted than the center.

$$\begin{aligned} AC_{searchwindow} &= G * S_{d^i_{width}} \\ TC_{searchwindow} &= \frac{1}{2} G * S_{d^i_{width}} \end{aligned} \tag{6.2}$$

The fragmentation scoring, F, within each search window is based on the frequency of unutilized slots and distance from the demand based on Equation 6.3, where e is the

Table 6.1: Table of Variables

Variable	Definition
n	Allocating node
i	Critiquing node
d	Physical Distance between nodes n to i
e	Length of the search window
s	Current search slot
x	Spectrum utilization ratio
U	Slot occupancy
C	Current demand
F	Current demand Fragmentation score
S	Search window total size
G	Current demand width
$width$	Spectral width

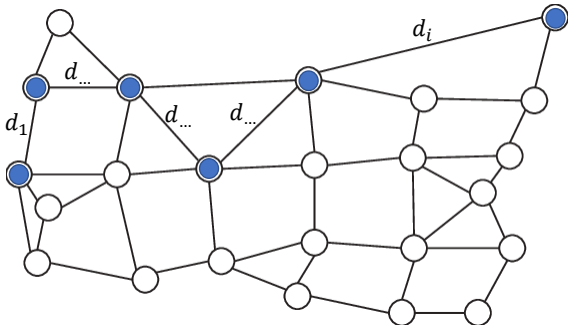

Figure 6.7: Scaling factor calculation

search window maximum length away from the demand, s is the current search slot starting at *zero*, x value capped at *one* representing utilization ratio, and the binary variable U is based on the state of the search slot to be used in conjunction with the scalar from Equation 6.1 to calculate the state for each demand.

$$F = \sum_{s}^{e} xUC^{e-s} \tag{6.3}$$

6.4.4 Training Variations

During the final stages of the TLFRL's development, we isolated several candidate approaches to best operate and train our RL model. Initially, the system is trained solely using the initial demand as seeds. Allowing unutilized slots to expand until the

total demand payload got depleted. To maintain correct critic scoring, variable x in equation 6.3 is used to block the inflated penalty resulting from the continuously emptying solution space by minimizing the fragmentation penalty, especially along the final hops preventing incorrectly trained RL actors. In addition to the training method above, two other variations are investigated.

- Treating all new demands as immutable without directly involving them in the training process.
- Fully integrating the new incoming demands upon arrival into the RL training alongside the initially seeded demands.

The most effective SA performance is achieved when new demands are treated as immutable. As a result, the system is able to reach a plateau more rapidly than when compared to integrating the demands, with comparable results discussed in the following section.

6.5 Performance Analysis

We require a comprehensive test environment setup using 5G demands services across isolated OTN long chains to accurately assess the performance of the proposed solution. The subsequent sections detail the specific aspects of the environment configuration utilized.

6.5.1 Generating Realistic 5G-Based Demands

To properly represent the dynamic nature of a number of research efforts generated simulation environment to that effect such as [93], We develop a realistic testbed to generate the necessary 5G-based demands using SUMO [94] to mimic a densely populated urban environment. Subsequently, the 5G traffic is managed using NS3, incorporating 5G-LENA built on it [95]. We customize NS3/5G-LENA to include core 5G services such as URLCC, mMTC, and eMBB-based demands like 8K video streaming, immersive gaming (including AR and VR), and tele-medicine. This ensures a highly diverse range of demands, from those with stringent latency requirements to those with heavy bandwidth needs.

6.5.2 Simulation Environment

The simulation and RL model are developed on a single workstation, operating sequentially, equipped with an i7-8700K processor, 32Gb RAM, and GTX1080TI GPU. The RL model is built and tested on PyQlearning Python reinforcement learning library, in conjunction with TensorFlow.

6.5.3 Test Scenarios and Metrics

To thoroughly evaluate the primary objectives of our system, the following test scenarios are conducted:

- Fragmentation ratios of the three system variations;
- Throughput
- Latency ratios of TLFRL with offloading allowed and blocked

6.5.4 Benchmarks

Two representative approaches are chosen to compare the performance of the proposed TLFRL scheme. The first is a MILP-based solution based on the work of Zhang *et al.* [96] applied on a fixed grid approach to show the advantages of the system's use of flex vs. fixed grid. The second benchmark is based on the heuristic portion from the work presented by Santos *et al.* [97] built on flexible-grid to represent current solutions adapted to the most recent spectrum setup changes in DWDM. The performance of our approach focuses on throughput against that of traditional survivability-focused SA approaches.

6.5.5 Results

Figure 6.8 shows the fragmentation results for all three variations of our solution. It is clear that there is a distinct advantage for the immutable approach in handling new demands, followed by the initial seeds only variation and, finally, the full integration model which exhibits clear shortcomings. The continuous re-feeding of the RL with new and neutral demands may explain this outcome, especially when the initial seeding

already provided comprehensive coverage of demand types and destinations. However, in the immutable approach, the addition of new demands enhanced the impact of the reward function, resulting in a slight improvement over the initial approach.

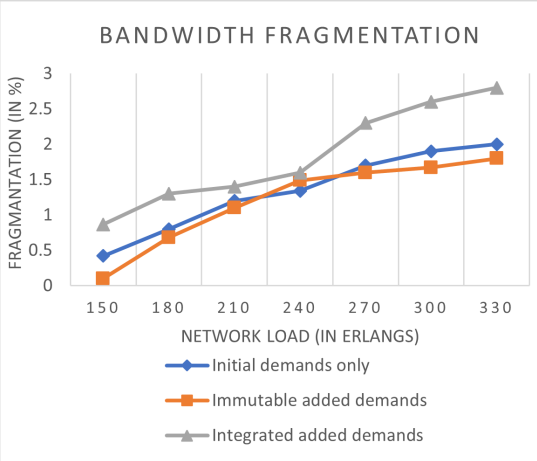

Figure 6.8: TLFRL's variations fragmentation comparison

Figure 6.9 highlights the throughput under increasing demand load. To more effectively assess this metric, all nodes are loaded with a cross-section of demand types comparable to the initial seed, minimizing the likelihood of size-based blocking and accurately representing high-demand periods. Our solution stands out from the accompanying benchmarks due to its sophisticated chain specific allocation technique, offloading capabilities, and optimal spectrum utilization. The fixed grid approach, which lags behind our solution, demonstrates the impact of proper spectrum allocation compared to the greedy-based approaches presented in the introduction section.

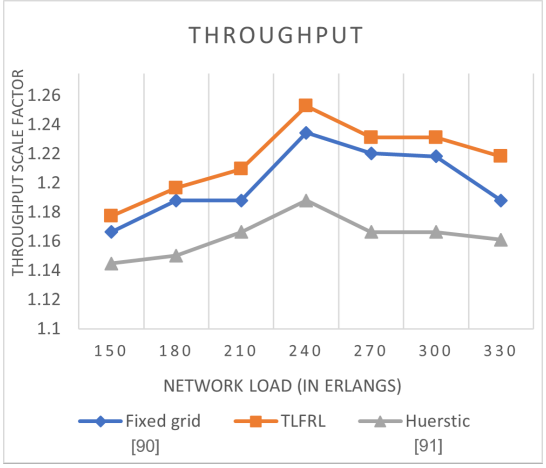

Figure 6.9: Throughput comparison

Figure 6.10 displays the latency violations that occur under progressively heavier traffic loads. For this test, in addition to the fully functional TLFRL, we include the disabled offloading variation to emphasize the minimal impact on latency caused by relying on traditional networking. We observe that the heuristic algorithm consistently underperforms compared to the other algorithms. The TLFRL algorithm and its variants maintain lower violation rates, albeit with small differences. Nonetheless, considering the minor latency differences into account compared to the throughput increase achieved by the TLFRL, this demonstrates the limited cost incurred to achieve the noted improvements.

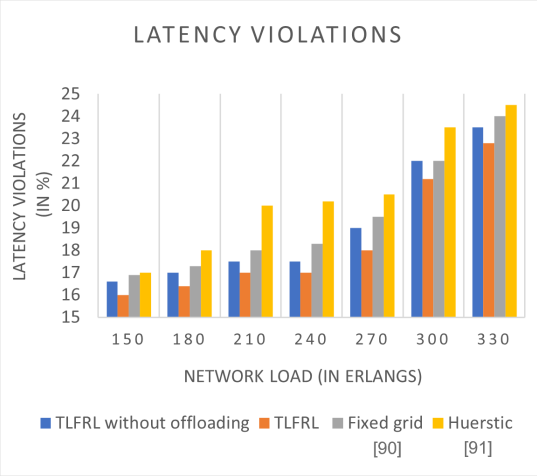

Figure 6.10: Latency violations comparison

Lastly, to check if using an RL-based approach will have an impact on the specific slots within the spectrum, Figure 6.11 illustrates the TLFRL's ability to offer a stable spectral assignment. The top row shows our system results in two clusters and has lower usage of the spectral window midsection. Compared to the heuristic approach in the lower row, it is observed that the placement is more haphazard, which can lead to more blocking probability with increased loads.

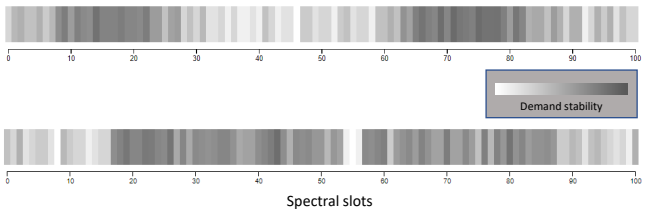

Figure 6.11: Spectral biases

6.6 Conclusion

In this chapter, we aimed to address the spectrum allocation problem in EONs while targeting improved throughput based on fragmentation while respecting latency constraints. We tackled the spectrum allocation process issues with several enhancements that enabled the RL model to train faster and achieve better results by dynamically reducing the granularity of the spectrum slots. Implementing immutable demands further reduced the solution space under average conditions. Additionally, we extended the spectrum allocation to traditional networking through an intelligent offloading scheme. Simulation results showed the system's minimal latency lag compared to the MILP and heuristic SA benchmarks, but with a significant increase in throughput.

In future research, we plan to extend our solution to accommodate multicast traffic grooming. We would also like to explore the implementation of time-of-day-based configurations of TLFRL to handle demand changes influenced by social factors.

Chapter 7
Transfer Learning-Accelerated Network Slice Management for Next Generation Services

7.1 Introduction

Next-Generation Networks (NGNs) are continuously being developed to improve networking performance and meet the users' ever-increasing demands. Three main services are currently the focus of 5G, namely Enhanced Mobile Broadband (eMBB), massive Machine-Type Communications (mMTC), and Ultra Reliability Low-Latency Communications (URLLC). Each service type requires a unique set of requirements to function properly. Given the rigid and closed nature of traditional Radio Access Network (RAN) solutions, hosting these types of demand in the future is impractical. The RAN has evolved through various changes to address its inherent challenges, including developing Centralized RAN (CRAN) and Virtualized RAN (vRAN). However, the lack of open interfaces and the reliance on proprietary hardware and software have hindered progress in making breakthrough improvements. the current research is tackling this issue on multiple domains such as edge computing [98], and 5G core [99] but remain hindered by the domains themselves. To overcome this Open RAN (O-RAN) upgrade has recently been introduced with much-needed flexibility and openness by utilizing virtualization, disaggregation, and most integral an open architecture [100].

Since its initial stages of development, O-RAN has been designed to enhance 5G systems by integrating virtualization elements along with Artificial Intelligence (AI) and Machine Learning (ML) techniques into its fundamental architecture. The introduction of two innovative modules, the near-Real-Time RAN Intelligent Controller (near-RT RIC) and non-Real-Time RAN (non-RT RIC). The Non-Real Time RIC is responsible

for handling the Orchestration and Automation functions in O-RAN. It manages radio resources, layer procedures, and policy optimization, focusing on integrating AI/ML models to support the operation of Near-Real Time RIC functions. This integration helps to improve guidance, parameters, policies, and overall stats within O-RAN. The Near-RT RIC, on the other hand, enables near-real-time optimization, control, and data monitoring of O-CU and O-DU nodes in live timescales (between 10 ms and 1 s). This is achieved by applying the policies generated by the AI/ML models computed/trained by the non-RT RIC. [101].

The demands of application and use cases in 5G networks are increasing in requirements and are expected to grow in volume for the foreseeable future. This led to a bottleneck in traditional reactive traffic management approaches attempting to meet these strict requirements of latency and reliability [102], especially when it comes to placement routing and traffic steering. To address this, Network virtualization was introduced to offer enhanced control over routing. Several solutions were developed to address these gaps effectively and to accommodate the dynamic nature of 5G better, focusing on different aspects of O-RAN. Among these, Intelligent Traffic Steering (TS) and Service Placement (SP) strategies have emerged as popular approaches [103].

Although TS and SP can effectively enhance the management of network functions and related aspects in O-RAN, their known limitations present a challenge in further improving their effectiveness for optimizing 5G demands in O-RAN. These limitations include the requirements to effectively orchestrate and coordinate various virtualized network components at individual levels. This complexity is further exacerbated due to the fluctuating nature of 5G demand patterns, widely differing networking requirements, and the necessity for immediate decision-making based heavily on accurate and timely data. In most cases, this data is limited in availability or quality, which inhibits the potential benefits of 5G O-RAN.

Although TS and SP significantly impact network functions management and related aspects in O-RAN, their inherent limitations, including lack of scalability, hinder the potential advancements in optimizing 5G demands in O-RAN. To overcome these limitations, proficient orchestration and coordination among various virtualized network components are necessary. Additionally, managing 5G demand is complex due to its dynamic nature, diverse service requirements, and immediate data-driven decisions, which

require timely, accurate, and reliable data. Unfortunately, the accuracy and timeliness of the required data are often limited by availability or quality issues, hence, hindering most of the sought after benefits.

Given the limitations mentioned above, it is evident that managing each demand separately is not a feasible approach. Thus, it is necessary to categorize and group the numerous demands based on their shared constraints within certain demand groups. Consequently, we focus on optimizing specific demand group constraints and achieving optimal utilization in a more efficient and effective manner. The concept of network slicing offers itself as the best-fitting tool to achieve our goal. This concept, gained widespread recognition when standardized in the 3rd Generation Partnership Project (3GPP) in 2018 [104].

This Thesis presents a zero-touch Network Slicing (NS) management solution comprised of three key elements: a Transfer Learning (TL) augmented Deep Q-Learning model, a repository for a local slice and RL agents, and, an Intelligent Controller heuristic as illustrated in Fig. 7.1. The solution aims to develop a set of agents and ready-to-use slices that are specifically designed for each O-RAN environment. The proposed solution first clusters the global demand set into groups based on three demand types and their constraints. Secondly, driven by the heuristic, the ML model creates agents for future slice instantiation and saves the agent for future use or transfer learning input.

The remainder of this chapter is organized as follows. In Section 7.2, we review the state-of-the-art literature. The motivation and problem definition are outlined in Section 7.3. The ML-based system's mathematical model and corresponding heuristic component are discussed in-depth in Section 7.4. Section 7.5 presents and discusses our simulation and its results. Finally, Section 7.6 concludes the thesis.

7.2 Related Works

Network slicing techniques is an area of interest for the research community, with the aim to improve the utilization and segmentation of 5G and next-generation demands [105]. However, there is a lack of focus on the management of O-RAN NS in general, beyond a specific use case. Most proposals in the literature are addressing specific cases

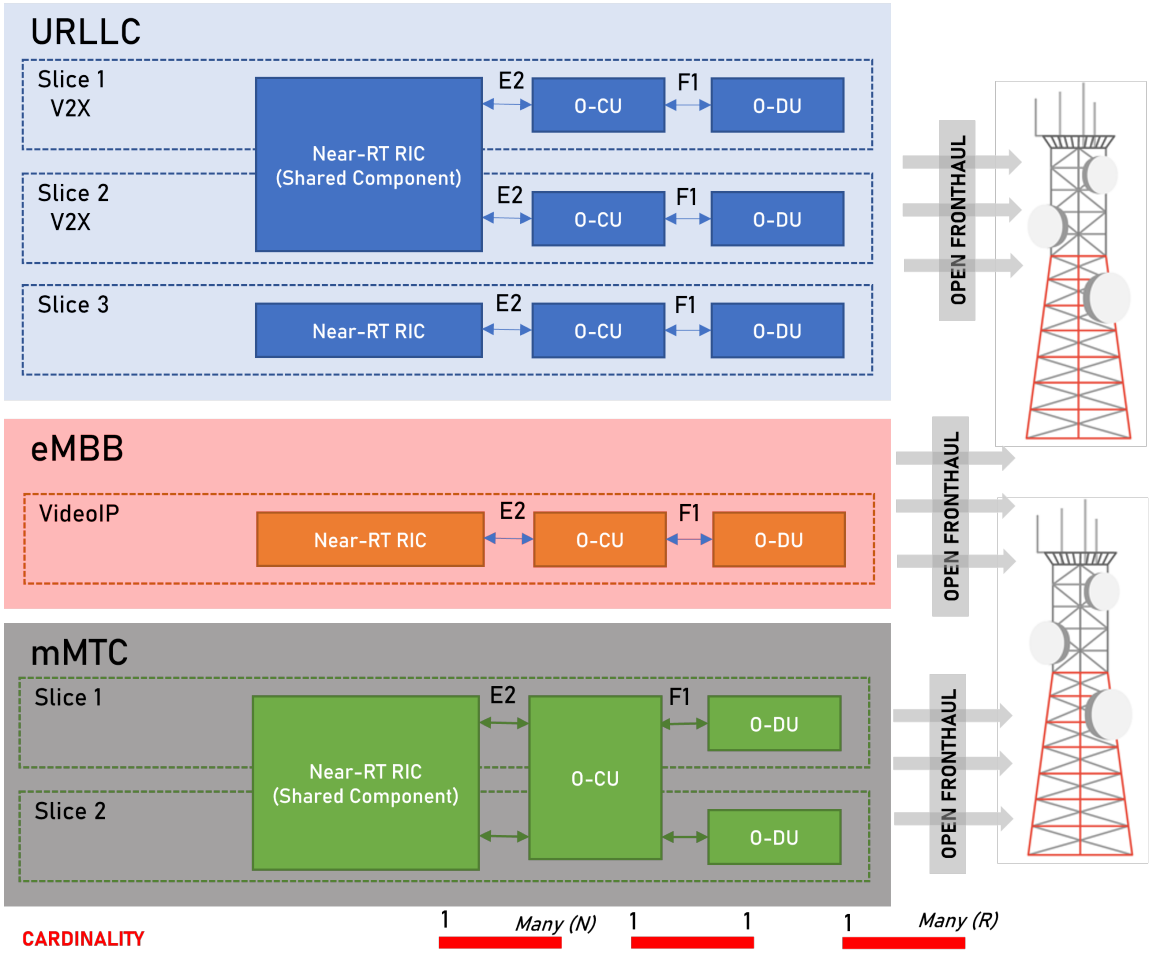

Figure 7.1: Network-slicing in O-RAN

in current O-RAN such as network function placement and traffic routing, which are NP-hard problems [106].

7.2.1 Optimization-based solutions

Due to the high complexity of the above-mentioned problems, optimization-based approaches became the next logical research tool. Motalleb *et al.* [107]tackled the challenge of improving the O-RAN overall performance by combining network slicing and power allocation strategies in an O-RAN system. To achieve this, The authors propose using a mixed-integer nonlinear programming (MINLP) framework to create a joint optimization model that optimizes resource allocations among various network slices while minimizing energy consumption. Their approach makes O-RAN more flexible and efficient in meeting cater to diverse service in future 5G and beyond network deployments.

Foroughi *et al.* [108] proposed a multi-objective optimization solution for NS in vRAN and O-RAN. The solution aims at optimizing multiple factors simultaneously, including quality of service (QoS), capital expenses (CAPEX), operational expenses (OPEX), and the transport network. The authors addressed O-RAN slice planning and design as a multi-objective binary optimization problem. The problem was solved using simulated annealing. Additionally, the authors introduced a slice-aware optimization model that aimed at minimizing CAPEX and OPEX while ensuring optimal quality of service. To do this, they took into account the transport network capacity, edge nodes, and front-haul delays. Their model achieved an average cost reduction of 70% within the tested network.

7.2.2 AI-based solutions

Yu *et al.* [106] presented an in-depth examination of the implementation and performance of dynamic 5G network slicing techniques across multiple network operators specifically for vehicular emergency scenarios. The authors designed and evaluated a multi-operator network slicing framework to dynamically allocate and manage network resources, and prioritizing emergency vehicular communication traffic. They evaluated the framework via simulation measuring key performance metrics such as latency, throughput, and resource utilization. The results showed that the proposed multi-operator network slicing framework significantly improved network performance and reliability by ensuring prioritized, low-latency communication for emergency services, particularly during high-demand emergencies. Their paper emphasized the potential benefits of their approach for enhancing real-time response and coordination during vehicular emergencies.

Thaliath *et al.* [109] explored the concept of predictive automation in the context of network slicing within an O-RAN. The authors' proposed a closed-loop predictive model that utilized ML algorithms to proactively optimize network resources. This model learns from historical network performance data to anticipate future service demands and adjust the network slices accordingly, ensuring optimal performance. The study showed that predictive approaches can significantly improve the efficiency and quality of service in an O-RAN system, enabling seamless adaptation to varying network conditions and user requirements.

Tamim *et al.* [110] chose to tackle the issue of network availability as the main goal and proposed an optimized deployment strategy for the virtualized O-RAN units in the O-Cloud to minimize the network's outages. Their Binary Integer Programing (BIP) model is optimized by employing placements to minimize the per-VNF and the SFCs' downtime and their redundant ones, maximizing the network's overall availability while adhering to the O-RAN-specific requirements.

Kak *et al.* [57] proposed an ML-based solution to forecast network traffic and optimally allocate radio resources across various network slices. Their aim is to enhance the effectiveness of network resource management. The solution Focuses on eMBB and mMTC slice types, emphasizing the importance of forecasting and subsequent slice reconfiguration. The testing results showed that the solution enables more autonomous, flexible, and responsive networking solutions, particularly in 5G and beyond network environments.

Duong *et al.* [111] tackled the constraints that come with the increased openness and virtualization of O-RAN especially when it comes to availability guarantees arising from either software or hardware failures. They created a decomposition model for the design of reliable O-RAN deployment under a dedicated virtual network function (VNF)-protection scheme. Their proposed model maximizes the network's yearly availability by providing a placement decision for all O-RAN VNFs and their backup instances. The model is solved by a column generation algorithm making it a scalable algorithm for large-scale 0-RAN deployments capable of producing near-optimal solutions in reasonable computational times regardless of the network scale.

Boateng *et al.* [112] leveraged the application of blockchain technology to spectrum trading in the context of NS in RAN to enable slice creation and autonomous slice adjustment. The authors also introduced a three-stage Stackelberg game framework for incentive maximization. A multi-agent deep reinforcement learning (MADRL) method is designed to achieve Stackelberg equilibrium (SE). The work simulation results demonstrate the effectiveness of the proposed method in terms of utility maximization and fairness.

Filali *et al.* [113] developed a two-level RAN slicing approach utilizing the O-RAN architecture to efficiently allocate communication and computation RAN resources among URLLC end devices. They formulated the resource slicing problem as a single-

agent Markov decision process for each RAN slicing level and used a deep reinforcement learning algorithm to solve it. Simulation results showed that the solution effectively met the quality of service requirements of URLLC.

Chergui *et al.* [114] presented an energy-aware Artificial Intelligence (AI) algorithm for Zero touch network and Service Management (ZSM) closed-loop automation. The algorithm uses statistical federated learning (StFL) to predict KPIs within a decentralized slicing architecture while adhering to stringent service level constraints. Compared to centralized constrained deep learning alternatives, and the proposed method exhibited significant improvements while also reducing the violation rate by a considerable margin in comparison to FedAvg. When analyzing many concurrent slices, the authors ensured the resulting solution would boast significant scalability and sustainability advantages over its predecessors.

Previous solutions shown effectiveness within their spheres but commonly lack scalability, particularly in relation to traditional optimization methods. Machine Learning (ML) solutions, while addressing certain network slicing (NS) challenges, they lack mobility and zero-touch capabilities due to their specialized focus. These limitations make integration into standalone O-RAN environments a challenge. Hence to overcome these drawbacks, we propose a solution leverages AI techniques to overcome these drawbacks. It's designed for independent deployment, adapting operations to maintain optimal 5G demand-based slices. This is achieved utilizing Reinforcement Learning and Transfer Learning using a unique reactive heuristic.

7.3 Motivation and Problem Definition

The mission-critical 5G applications such as Automatic Vehicle Locator (AVL) for task-sensitive vehicles and latency/reliability-based applications such as Augmented Reality (AR)-assisted surgery, Vehicle-to-Everything (V2X) communications, live sports events streaming, and online gaming are the leading areas that benefited from the 5G revolution. These applications are at the forefront of further development in Beyond 5G (B5G) and 6G networks under the use case of URLLC. However, network traffic and user connectivity are expected to grow rapidly in the upcoming years as the number of

mobile 5G subscriptions worldwide is expected to reach 4.8B by 2026 and pose further challenges for these networks [77].

The 5G revolution and projected advancements in Beyond 5G (B5G) and 6G networks provided faster and more reliable wireless communication, making Internet connectivity a ubiquitous feature of everyday life as initially highlighted in [115] focusing on the heterogeneity and resulting interference along with their impacts. the work was expanded on recently in [116] critiquing a number of solutions that have been introduced since. Along with all the challenges the impact of such services has become evermore multifaceted and integral across various critical sectors, such as:

- Internet of Things (IoT): With enhanced connectivity, IoT expands support to vast number of connected devices, enabling the advancement of smart homes, cities, factories, and agriculture.
- E-Health: Facilitating the realization of remote patient monitoring, telemedicine, and even remote surgery. Advanced healthcare technologies like AI diagnostics and personalized medicine become more stable, accessible, and efficient.
- Industry 4.0: Enabling real-time data sharing, remote machinery control, and increased automation resulting on more efficient and flexible factories.
- Autonomous Vehicles and Drones: require reliable, fast, low latency, and high capacity communication.
- AR/VR: Expedite the advancements in Augmented Reality (AR) and Virtual Reality (VR), enabled more immersive experiences and practical applications in areas like education, training, and entertainment.

To distinguish between such a highly divergent services, 5G categorized its applications into three major categories based on their specific requirements for optimal performance. These categories are:

1. eMBB covers services that need high data-rate transmissions and provide enhanced broadband experiences. This includes 4K/8K video streaming VR or AR.

Table 7.1: URLLC use cases

Type	**Description**	**Example**
Low Latency with High Reliability	Applications Requiring very low latency (on the order of milliseconds) and extremely high reliability (up to 99.999%)	Autonomous vehicles, factory automation, and remote surgery.
Low Latency with Moderate Reliability	Applications that tolerate slightly lower levels of reliability, but the need for quick response times remain, with occasional data packet loss not having critical consequences.	AR/VR applications and online gaming.
Moderate Latency with High Reliability	Applications requiring highly reliable communication to maintain stability, but the acceptable latency could be slightly higher than in life-critical applications.	Smart grid and fleet Management
Low Latency with High Data Rates	Applications requiring high data rates in real-time.	ITS and remote control of heavy machinery

2. mMTC focuses on supporting large scale, low-cost, low-energy consumption applications mainly associated with the Internet of Things (IoT), including Smart meters, Asset tracking devices, and Connected agricultural sensors.

3. URLLC targets applications requiring very low latency and ultra-high reliability, such as Autonomous vehicles, Remote surgery, and Industrial automation and control.

Managing these services can be complex, especially because each category has informal subdivisions. For instance, URLLC can be divided based on the variation in constraints it has, as shown in table 7.1. These challenges are most noticeable in the front to mid-haul regions due to their sporadic placement. Therefore, there have been significant academic and industrial efforts to address them, including the development of O-RAN, one of the leading platforms tailored toward 5G integration.

As the O-RAN architecture enables the utilization of AI and ML across all its layers, leading to more intelligent solutions and improved performance in shorter time than primitive heuristic-based solutions . However, to take advantage of this, it is crucial to continuously optimize the resource allocation for services within O-RAN. To achieve

this, it is paramount to ensure that the 5G related O-RAN components, mainly the near-RT RIC, O-RAN Centralized Unit (O-CU), and O-RAN Distributed Unit (O-DU), are utilized efficiently based on the combination of active 5G applications.

One main aspect differentiating O-RAN from its RAN predecessors is its full virtualization. Although this offers several benefits, it also presents a set of challenges due to the varied inter-component latencies and capabilities. Given this new dimension of complexity, placement techniques that deal with applications individually are bound to reach a bottleneck due to scalability limitations and processing time. A more efficient method of overcoming these challenges is to process the 5G demands in bulk; a well-known method capable of such a task is network slicing. Network slicing divides the physical network into multiple virtual ones, called slices. Each slice can be independently managed and optimized to suit our specific application type and its constraints, allowing for more efficient use of network resources and improved performance.

This work presents an extended zero-touch scalable solution within the O-RAN architecture. The system is comprised of four main components: a heuristic algorithm to control and automate the other three components to allow for zero-touch capabilities, a Deep Reinforcement learning (DRL) model as the primary slice creation tool, a transfer learning model to improve the outcomes and processing time of the RL model, and finally, a repository holding the network slices and trained RL agents for creating network slices and training additional agents, ultimately ensuring successful deployments as shown in Fig. 7.2.

The DRL is used to create the agents capable of slicing the network for each set of grouped 5G applications optimizing both latency and reliability. These agents are trained on three distinct sets of constraints to account for the differences in 5G applications discussed earlier. The heuristic algorithm handles the storing of slices and agents, triggering new agent creation, or activating stored agents based on incoming applications and their similarity to the trained RL agent's input.

To evaluate our end-to-end solution, we build a large-scale O-RAN environment and utilized real 5G datasets to stress test the developed solution properly. This ensures it is future-proof under projected increasing scale and dynamic network conditions.

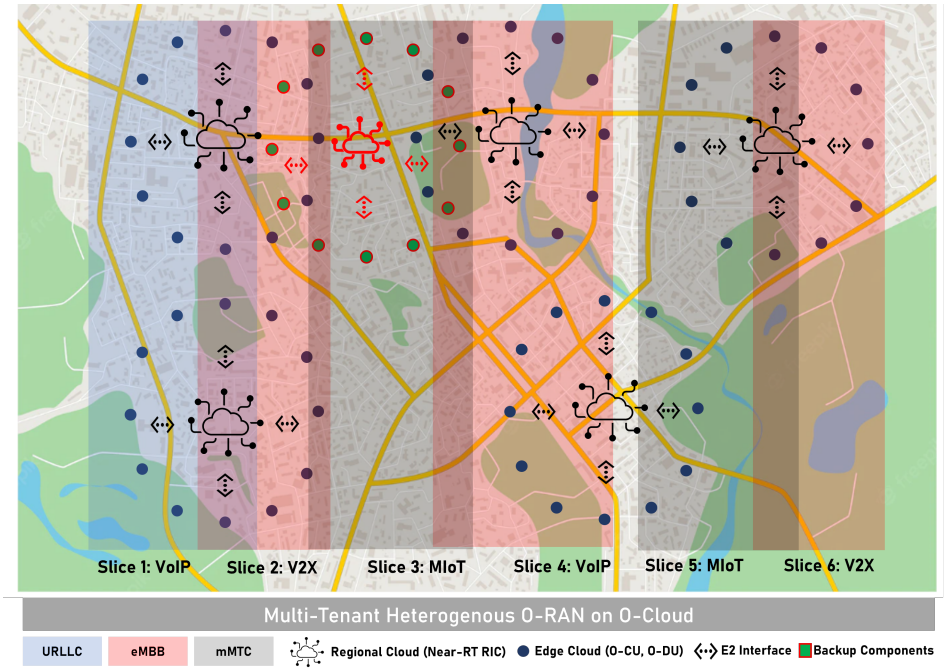

Figure 7.2: System Overview

7.4 System Description

The system's composed of two core modules: the ML engine illustrated in Fig. 7.3, and the heuristic module. The ML engine consists of a Deep reinforcement learner augmented by a compatible transfer learning model. This is triggered by the heuristic module output. The heuristic module matches the correct stored agent to reduce the training cycle length and ensure consistent performance during the deployment period. The system's repository is controlled by the heuristic module and used to create the slices when a capable agent is available, then store both newly created agents and resulted networking slicing decision. The system core can handle the three 5G demand types: URLLC, eMBB, and mMTC. The following sections outline the DRL model formulation and the heuristic algorithm.

7.4.1 Machine learning component

The engine's goal is to optimize the placement location of all O-RAN VNFs and their one-to-one backup components, ensuring maximum availability of the deployed slice. To achieve this, an optimization problem is formulated with the objective function to minimize the downtime incurred by the VNFs hosted on the commercial-off-the-shelf

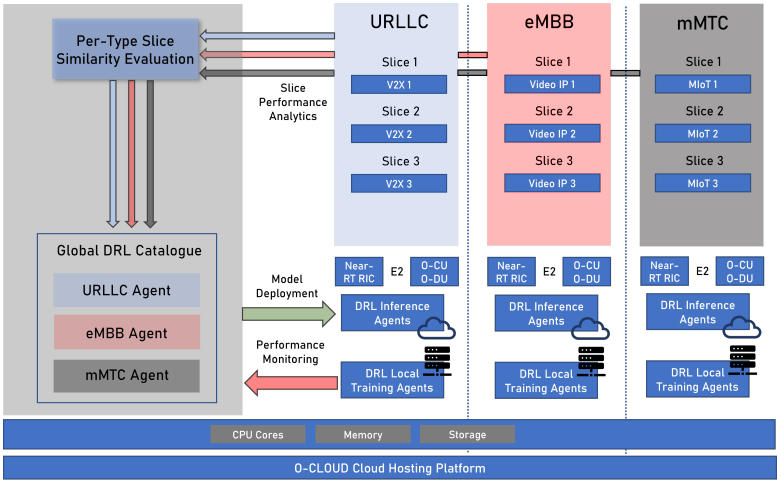

Figure 7.3: TL augmented RL System

(COTS) servers. The engine receives a request from the heuristic to create an O-RAN slice with a required number of VNFs, a set of candidate servers available to the network service provider (NSP), and candidate agents to apply transfer learning where possible. The engine aims to place all the VNFs and their backup components to minimize the overall downtime across all SFC as shown in Eq. 7.1. This objective applies to all three use cases. Each use case has specific constraints that are tailored to the engine. These constraints are outlined in the following sections.

The objective function, Eq. 7.1, aims at minimizing the total downtime across all hosted VNFs. The downtime of a VNF on a server is calculated by dividing its recovery time (RT) by its Mean Time to Fail. This same calculation is used to determine the downtime of a server. The sum of these values represents the downtime of a VNF hosted on a specific server.

$$\min \sum_{servers} \sum_{VNFs} \left[\left[\frac{RT_{vnf}}{MTTF_{vnf}} + \frac{RT_{server}}{MTTF_{server}} \right] \right] \times \theta_{vnf,server} \quad (7.1)$$

Each use case for O-RAN has its own unique operational and functional constraints. Our ML engine considers these on a per-use case basis to ensure that any deployment issued is tailored correctly. For URLLC deployments, we focus on are optimizing reliability while strictly adhering to latency. For eMBB deployments, we consider the data transfer capacities of all the servers and maximize the SFC availability. Finally, for mMTC deployments, we aim at efficiently serve as many end-user equipment as possible while

meeting the minimum number of servers set by the NSP. The equations provided below list the specific constraints for each type of deployment.

7.4.1.1 eMBB constraints

Eq. 7.2 ensures that server *"server"* is capable of providing uplink data rates that are equal to or exceed the uplink data rates threshold set for eMBB for all hosted VNFs *"vnf"*.

$$\theta_{vnf,server} \times UL_{server} \geq UL_{\eta} \times \theta_{vnf,server} \forall vnf \in VNFs, \forall server \in Servers \quad (7.2)$$

Eq. 7.3 ensures that server *"server"* is capable of providing downlink data rates that are equal to or exceed the downlink data rates threshold set for eMBB for all hosted VNFs *"vnf"*.

$$\theta_{vnf,server} \times DL_{server} \geq DL_{\eta} \times \theta_{vnf,server} \forall vnf \in VNFs, \forall server \in Servers \quad (7.3)$$

To ensure that the links between the SFC VNFs meets the bandwidth requirements for O-RAN eMBB use cases, Eq. 7.4 ensures that the bandwidth of all the links of the hosting server is equal to or greater than eMBB bandwidth threshold.

$$\theta_{vnf,server} \times BW_{server} \geq BW_{\eta} \times \theta_{vnf,server} \forall vnf \in VNFs, \forall server \in Servers \quad (7.4)$$

As part of the operational constraints of O-RAN, Eqs. 7.5, 7.6, 7.8, 7.9, 7.11, and 7.12 ensure that no VNF can be hosted on a server that does not have enough CPU cores or memory capacity. Each use case can have the required CPU and memory set by the NSP. Hence, each use case has a different threshold as shown in the equations.

$$\sum_{v=0}^{VNFs} \sum_{s=0}^{Servers} \varsigma_{v_{eMBB}}^{req} < \varsigma_{s}^{avail} \times \theta_{v,s} \quad (7.5)$$

$$\sum_{v=0}^{VNFs} \sum_{s=0}^{Servers} \xi_{v_{eMBB}}^{req} < \xi_s^{avail} \times \theta_{v,s} \quad (7.6)$$

7.4.1.2 URLLC constraints

The core constraint of URLLC is to ensure that the links between each dependent VNF have a latency less than or equal to the latency threshold set by O-RAN URLLC deployment specifications. Eq. 7.7 depicts this constraint showing an iteration of all VNFs and servers, but only considering servers that have VNFs hosted on them.

$$\sum_{v=0}^{VNFs} \sum_{s=0}^{Servers} \sum_{v_d=0}^{VNFs} \sum_{s_d=0}^{Servers} l_{s,s_d} \geq l_{v,v_d}^{threshold_{URLLC}} \times \theta_{v,s} \times \theta_{v_d,s_d} \quad (7.7)$$

$$\sum_{v=0}^{VNFs} \sum_{s=0}^{Servers} \varsigma_{v_{URLLC}}^{req} < \varsigma_s^{avail} \times \theta_{v,s} \quad (7.8)$$

$$\sum_{v=0}^{VNFs} \sum_{s=0}^{Servers} \xi_{v_{URLLC}}^{req} < \xi_s^{avail} \times \theta_{v,s} \quad (7.9)$$

7.4.1.3 mmTC constraints

For mMTC, Eq. 7.10 improves efficiency by reducing the needed number of hosting servers. The NSP can set a parameter called ς to limit the agent to using no more than ς servers.. If no solution is possible with ς servers, the engine will incrementally add more candidate servers while notifying the NSPs until a solution is found. This constraint enables NSP to increase servers efficiency and hence serve more UEs per location on a massive scale using the same server infrastructure. However, this comes at the cost of computational resources, which NSPs can control by adjusting ς.

$$\sum_{\partial=0}^{|SFCs|} S_{\partial}^{UniqueCount} \leq \varepsilon_{mMTC}^{threshold} \quad (7.10)$$

$$\sum_{v=0}^{VNFs} \sum_{s=0}^{Servers} \varsigma_{v_{mMTC}}^{req} < \varsigma_s^{avail} \times \theta_{v,s} \quad (7.11)$$

$$\sum_{v=0}^{VNFs} \sum_{s=0}^{Servers} \xi_{v_{mMTC}}^{req} < \xi_s^{avail} \times \theta_{v,s} \quad (7.12)$$

7.4.1.4 Transfer learning

For improved accuracy and faster deployment, we decided to incorporate a TL component to enhance our RL. We used transfer learning's domain adaptation technique to leverage agents that were already trained based on earlier slice requirements and O-RAN data collected from Non-real-time RIC (source domain). This allowed us to adapt to a new set of slice requirements and O-RAN data (target domain), as shown in Eq. 7.13. With x^t and y^t representing the input data and labels from our historical data. We chose this method because it is the most straightforward and effective approach for our environment, where the distribution of data in the target domain is only slightly different from the source domain, which is often the case in real-world networking deployments.

$$\theta^*_{\text{target}} \quad <<< \quad \min_{\theta} \sum_{(x^t,y^t)\sim D_t} [L(M(x^t;\theta), y^t)] \quad (7.13)$$

7.4.2 Heuristic component

Although the RL can efficiently create slices and agents, it still cannot be considered as a zero-touch solution as it still requires an intelligent driver. To address this, we developed a heuristic solution to trigger agents and slice creation at the DRL. Fig. 7.4 outlines the operational steps of the heuristic algorithm. The heuristic algorithm as the system driver covers the entire process, from the initial scenario of an empty repository to the eventual stability where the need for ML becomes negligible.

Algorithm 5 shows the triggering methodology adopted by our heuristic along with automated adjustment of the agent creation rate from the initial stages till the system reaches a stable state. This is necessary because, in the earlier stages, more agents are

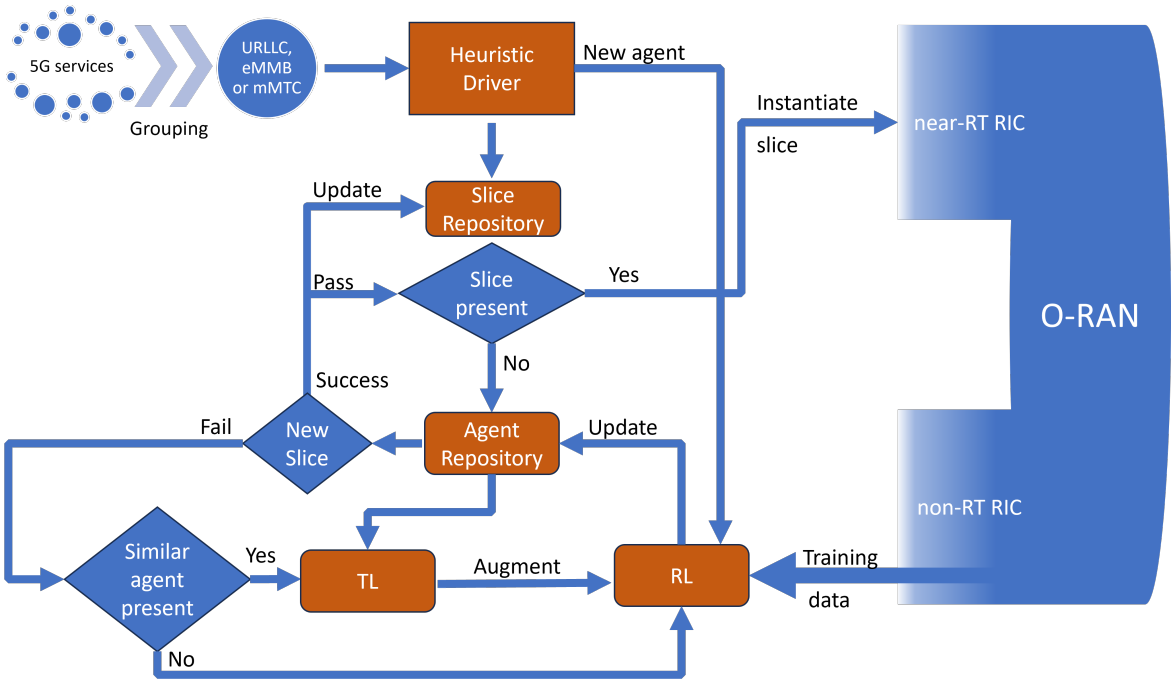

Figure 7.4: Heuristic Driver overview

Algorithm 5 Heuristic driver.

Require: $S_{similarity-index}$,$S_{similarity-threshold}$; $A_{similarity-index}$,$A_{similarity-threshold}$; $A_{learning-threshold}$

New incoming 5G demand set
if $S_{similarity-index} > S_{similarity-threshold}$ **then**
  Use stored slice image
  Check slice Fit
  if Stored Slice creation fails or Fit fails **then**
    if $A_{similarity-index} > A_{similarity-threshold}$ **then**
      search for the best-fit agent to create a new slice
      if $A_{similarity-index} > A_{learning-threshold}$ **then**
        Create and Store the new slice
        Update $A_{learning-threshold}$
      end if
      Trigger new agent creation
    end if
  end if
end if
Engage RL
if $A_{similarity-index} > A_{learning-threshold}$ **then**
  Augment with TL
  Update $A_{learning-threshold}$
end if

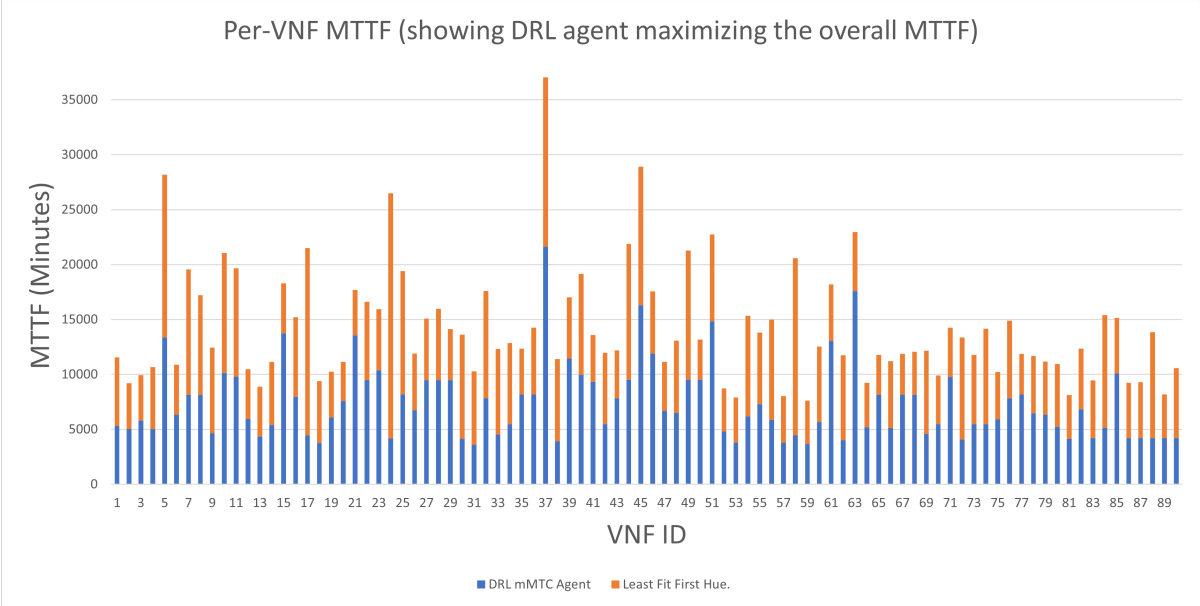

Figure 7.5: MTTF Across All Placed VNFs for mMTC Application

needed to improve the impact of the transfer learning part of our model. Later the rate of agent production is reduced to focus on direct slice implementation. The variables $A_{similarity-threshold}$ represent the similarity threshold used to trigger a new learning cycle vs. enforced direct slicing using the existing agents, while $S_{similarity-threshold}$ is used as a bias adjuster that is updated after each cycle to shift operation away from creating new agents based on the robustness of the agent and slice repositories. The size of the repository is important especially when tacking model drift-related loss of accuracy typical in such scenarios [117].

7.5 Performance Evaluation

Our simulation setup consists of the O-RAN environment, key metrics outlined in table 7.2, and 5G application-based generated datasets. To evaluate our proposed heuristic-driven ML engine, we deployed our agent in an environment with the incoming requests for each 5G application type as packets per second from UEs to every VNF for a one day period (1440 timesteps) through a discrete event simulator, following the specifications presented in [118]. The slice failure at any VNF is tracked over a period of several months.

The chart shown in Fig. 7.5 demonstrates how our solution performance compares to the traditional individualized approach algorithm. We used the MTTF per VNF in

Table 7.2: O-RAN Environment and Metrics Description

Attribute	**Description**	**Distribution**
VNF ID	Unique ID of VNFs available	Range 0-99
VNF types	List of O-RAN component types	Near-RT RIC, O-CU, O-DU
RT	Recovery time	Normal distributed random variable, Mean 0.008, Var. 0.005
CPU	Number of Cores available for each component	Near-RT RIC: 8 O-CU, O-DU: 4
Memory	Available RAM in GB	Near-RT RIC: 16, O-CU, O-DU: 6
MTTF	Mean time between failures	Exponential distributed random variable, Mean 2100
MTTR	Mean Time To Recovery	Normal distributed random variable, Mean 0.05, Var. 0.03
DL	Downlink GB	Uniform distributed random variable, Range 17-35
UL	Uplink GB	Uniform distributed random variable, Range 8-20
Bandwidth	specifies the available bandwidth in (GHz)	Dist. Uniform, 1-3

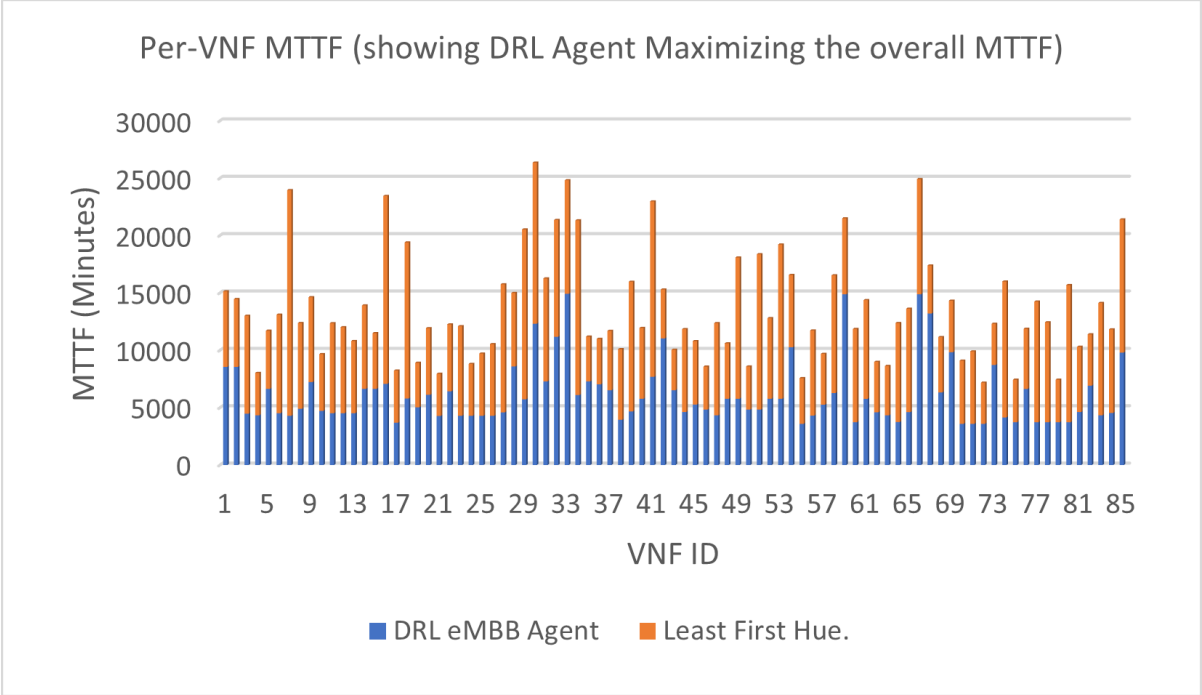

Figure 7.6: MTTF Across All Placed VNFs for eMBB Application

minutes for mMTC slices during the testing period as the comparison metric. Based on the chart, it highlights the advantages our approach has produced across most VNFs. The 89 VNf snapshot shows our solution, while dealing with placement in bulk, maintained its advantages compared to the benchmarking algorithm working on an individual based, giving it more opportunity for optimization, but given its approach to the placement of the three chains more often than not it creates less optimal placements; this can be attributed to greedy approach in the second hop particularly resulting in a bad list of choices for the final chain especially when the latency requirements constraints limit it to adjacent units only.

In Fig. 7.6, it can be seen that the eMBB simulation results show comparable results mMTC. This confirms the advantages of our approach of segregating the slices and agents based on application types and treating each slice based on its unique traits.

Fig. 7.7 on the other hand, has shown simulation only slight improvements compared to the other two service types. This is due to the URLLC service restrictive constraints outlined in the section above, which resulted in limiting our slice capabilities when compared to the individualized approach. while these results were expected there is room for improvement discussed in the conclusion section that we aim to explore.

even with the limitations imposed by our grouped bulk processing, our solution

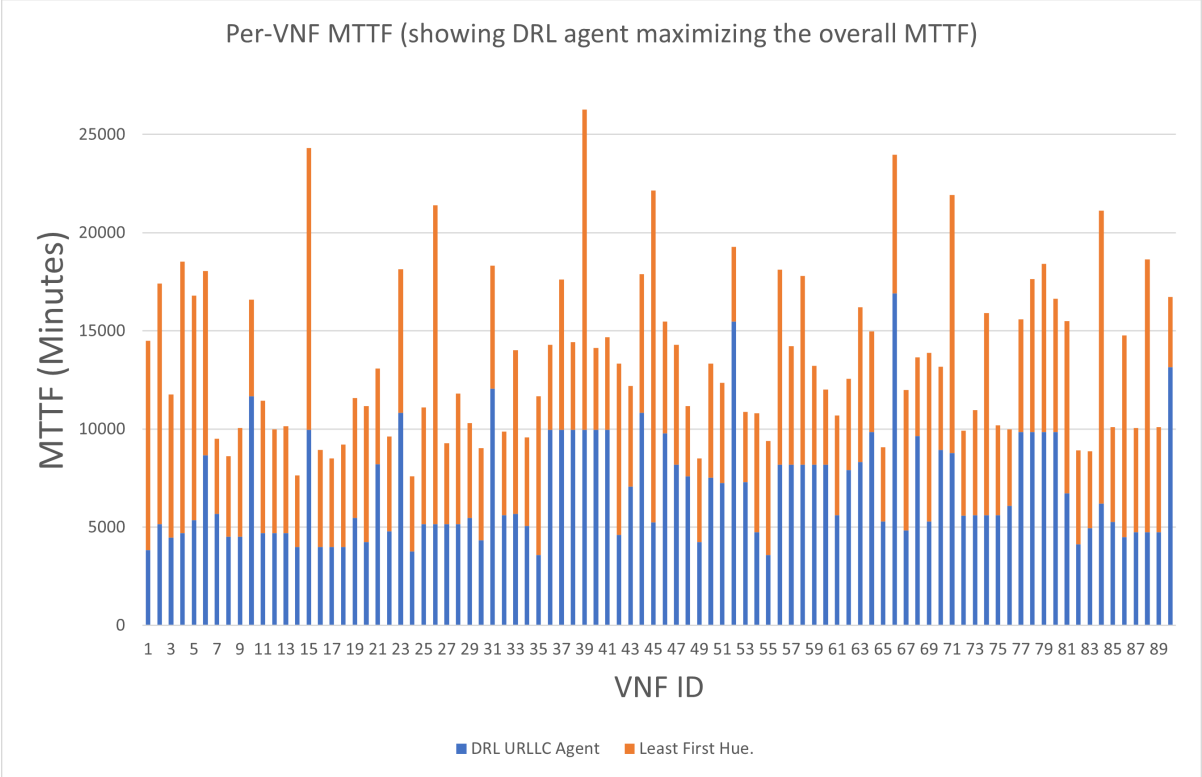

Figure 7.7: MTTF Across All Placed VNFs for URLLC Application

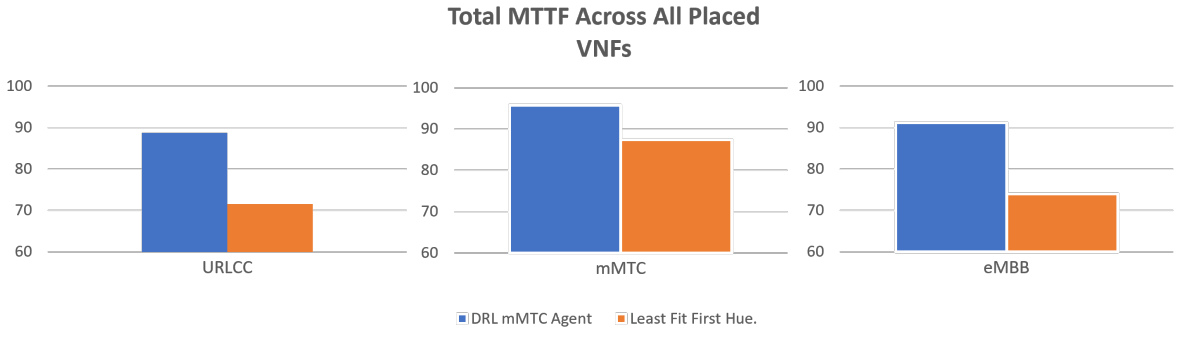

Figure 7.8: Impact of TL on Agent training

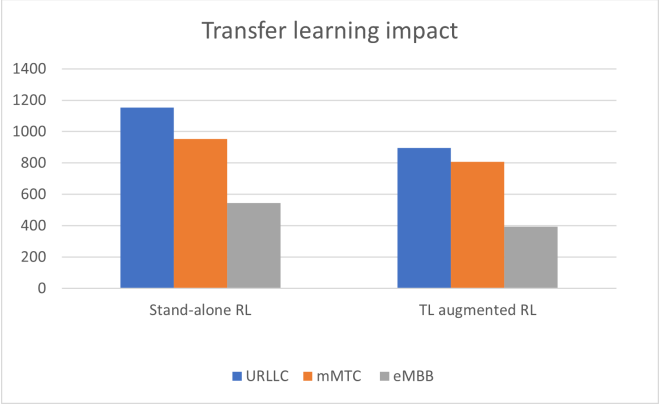

Figure 7.9: Impact of TL on Agent training

remains consistent in it targeted improvements with an 11.31% overall increase in MTTF, as illustrated in Fig. 7.8. These findings highlight the benefits of using NS deployment and management over traditional approaches regarding robustness, fault tolerance, and recovery. even in URLCC maintained, such improvements across the testbed were with approximately a 5.16% increase.

Finally, we aimed to validate the impact of TL on the proposed RL. To achieve this, we compared the performance of our TL-augmented setup to the standalone RL, as illustrated in Fig. 7.9. Using the TL in its basic form still results in a significant decrease in training time, especially in eMBB. This reduction in the training time is advantageous as it allows for more frequent agents update, which in hand, will reduce the amount of time it takes for our model to achieve stability regardless of the O-RAN deployment scale.

7.6 Conclusion

We presented a zero-touch solution for managing 5G network slicing. Our solution uses a heuristic-guided TL-augmented RL implemented within O-RAN. the solution has effectively minimized downtime while still complying with O-RAN's operational and functional limitations. The solution is highly adaptable and can easily accommodate sudden changes to the deployed network. The system uses unique and locally available information, facilitating easier integration. The performance of the proposed system is evaluated in comparison to traditional approaches using industry-standard 5G traffic datasets. The evaluation results show that the proposed system consistently achieves

lower downtime than the traditional algorithms. For future work, we aim to expand the RL environment by incorporating additional state parameters to enable more effective deployment of larger O-RAN. additionally, we would like to investigate the approach of using the constraints of the requests directly instead of grouping based on type first, resulting in heterogeneous slices when it comes to the types occupied but more aligned constraints, which theoretically should produce better optimally despite the slight increase in complexity. As for the TL, we have begun retooling the portion of our system to use Partial Fine-tuning approaches as well as investigate Transfer learning normalization techniques to improve the augmentation in both production speed and accuracy. Finally, we intend to implement our solution in real-world O-RAN deployments to better test its performance under a larger set of URLLC traffic.

Chapter 8
Conclusion

8.1 Introduction

The continued evolution and penetration of technology in our daily lives has led to a rapid acceleration in their application and networking requirements. More and more enterprises and businesses are adopting 5G-based services such as URLLC, eMMB, and eMTC. However, adopting such technologies required high virtualization of the networking structure that brought many enabling features but also introduced several challenges, such as the urgent need to improve the performance and efficiency of such systems, which may be achieved by properly placing the available services, especially within the scarce resources typical in the edge computing environment. Another non-trivial challenge is optimizing the use of the highly stable and massive throughput of OTN networks to better support 5G services. A third challenge is managing 5G services within O-RAN, especially with the increased dynamicity of the demands. To that end, this thesis proposed using various optimization modeling and machine learning techniques in three different domains; Edge computing (front-haul), O-RAN (Mid-haul), and OTN (backbone). In particular, the first part of the thesis focused on using optimization modeling techniques to improve downtime through orchestration. The second part of the thesis created a simulation environment for 5G in OTN and trained an RL model to enhance the spectrum allocation scheme towards lower fragmentation and better throughput. The third part of the thesis proposed employing Network slicing using TL-augmented RL to achieve zero-touch 5G optimization in O-RAN.

The remainder of this chapter summarizes the contributions achieved, the schemes developed in this thesis, and the probable future research directions.

8.2 Summary of Contributions

Chapter 3 formulated the container orchestration problem within an edge computing environment as a Mixed Integer Linear Programming (MILP) to minimize downtime. While effective at orchestration, the complexity of the solution made it best suited for benchmarking. The problem was then reduced further into a heuristic scheme to enable practical implementation. Results showed that the optimal solution achieved a global minimum, and the heuristic scheme achieved lower downtime unhindered by the reduction in solution space even across the various testbeds with different densities. Chapter 4 extended the previous work by introducing a better segmentation scheme for edge computing space to reduce the solution space and allow for local orchestration. Due to the containers' stringent demands coupled with users' dynamic nature within an edge environment, grid or static methods are inefficient. Hence, the proposal of lax clustering techniques as a tool to create the subspaces. Chapter 5 utilized various simulation tools to create a modular 5G OTN simulation environment. Due to its distributed nature, the resulting solution could create large datasets with minimal equipment. Chapter 6 proposed using reinforcement machine learning techniques to better place 5G flows on the spectrum within isolated optical paths. The work in Chapter 6 achieves lower fragmentation and decreased blocking, leading to better throughput. In Chapter 7, the topic of 5G placement within O-RAN was addressed to reduce complexity. The 5G applications were categorized into three main types. Heuristic and machine learning techniques, mainly a TL-augmented RL engine, were used to create customized network slices. The solution achieved Positive results the demands being grouped and processed in bulk.

8.3 Future Research Directions

This thesis explored the use of optimization modeling and machine learning techniques to automate systems more efficiently and intelligently, with each of the three solutions developed showing impactful results in their respective target. Despite the novel solutions and algorithms that were developed in this thesis to tackle the various challenges of B5G zero-touch systems, several challenges still need to be addressed. These challenges are of technical, economical, regulatory, and social nature, given the multi-

national, multi-industry, and multi-technology ecosystem that the zero-touch systems will create. This section presents future research directions that can be explored toward more efficient and intelligent systems, with specific examples for each of the three systems considered in this work.

8.3.1 Technical Challenges:

Many technical challenges need to be addressed in 5G-based systems, which offer numerous opportunities for researchers to explore. Some of these challenges include complexity, management overhead, and reliability, to state a few. Researchers can use various techniques and methods to overcome these challenges, such as data analytics, Deep learning, optimization, and game theory. Here are three examples of potential future research directions that can build on the work presented in this thesis.

8.3.1.1 Edge Computing Segmentation and Container Orchestration

The research discussed in Chapters 3 and 4 can be expanded in various ways. One potential expansion is to include more metrics like container hierarchy, Kubernetes backups, and other management systems and consider their role in the decision-making process for orchestration. Furthermore, examining the energy efficiency of the process and triggering mechanism could lead to greater energy savings for service providers and devices while maintaining performance and reducing the frequency of orchestration cycles.

8.3.1.2 Optical Simulation and Spectrum Allocation

The work presented in Chapters 5 and 6 can be extended in several research directions. To better simulate the environment of 5G and OTN, the granularity of OTN paths implementation can be improved using state-of-the-art ROADMs and other unique optical networking components to better simulate the optical aspect of the networks. Another possible extension is using advanced classification techniques to detect and forecast the flows to proactively allocate the spectrum achieving better stability for OTN transmission despite the projected increased heterogeneity of B5G and NGNs.

8.3.1.3 Network Slicing in O-RAN

Chapter 7's work can be expanded by exploring various research directions. One such direction is to divide the NS agents based on more attributes like latency requirements and security demands. This can enhance the NS allocation process. Another potential extension is to refine the heuristic scheme for identifying O-RAN states where it is better to redeploy full NSs instead of creating them using the existing agents working solely on remaining available resources.

8.3.2 Economical Challenges:

When creating Zero-touch systems, addressing the economic challenges they present is important. Advanced machine learning techniques should be used to create efficient models to reduce both CAPEX and OPEX. Additionally, virtualization technology can be used to share physical infrastructure and equipment among multiple vendors. For B5G zero-touch systems, new and innovative business models are necessary, as traditional models are ineffective in the complex multi-industry and multi-technology ecosystem they create. Game theory can be utilized to develop these models, as it can handle the diverse nature of B5G systems.

8.3.3 Regulatory Challenges:

Another challenge is the regulatory and standardization process for future zero-touch systems. This is crucial given that such standards are needed to determine the application requirements, communication protocols, security practices, and other operation mechanisms for an efficient zero-touch system. To that end, the different standards development organizations such as the Internet Engineering Task Force (IETF), International Telecommunication Union (ITU), and 3rd Generation Partnership Project (3GPP) need to develop and ratify the standards to govern B5G systems. For example, the communication standards needed to enable and facilitate 6G communication within a heterogeneous network setup.

8.3.4 Social Challenges:

The social aspect of zero-touch systems is crucial due to concerns that they may replace humans, particularly in sectors like network engineering. In this field, it's important for developers and technicians to treat patients as individuals rather than just statistics. Skeptics worry that standardized approaches could be a weakness, relying too much on a one-size-fits-all approach rather than traditional techniques. Therefore, it's essential to create intelligent systems that act as support tools rather than complete replacements for human actions and interactions.

References

[1] A. Machen, S. Wang *et al.*, "Live service migration in mobile edge clouds," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 140–147, 2018.

[2] grand view research. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/5g-services-market

[3] S. Pongratz, "The role of intelligent ran and automation," Jun 2022. [Online]. Available: https://www.delloro.com/the-role-of-intelligent-ran-and-automation/

[4] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Edge-enabled v2x service placement for intelligent transportation systems," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1380–1392, 2021.

[5] ——, "Cost-optimal v2x service placement in distributed cloud/edge environment," in *2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2020, pp. 1–6.

[6] A. Moubayed and A. Shami, "Softwarization, virtualization, and machine learning for intelligent and effective vehicle-to-everything communications," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 2, p. 156–173, 2022. [Online]. Available: http://dx.doi.org/10.1109/MITS.2020.3014124

[7] O. Sefraoui, M. Aissaoui, M. Eleuldj *et al.*, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.

[8] B. I. Ismail, E. M. Goortani *et al.*, "Evaluation of docker as edge computing platform," in *2015 IEEE Conference on Open Systems (ICOS)*, 2015, pp. 130–135.

[9] U. Awada and J. Zhang, "Edge federation: A dependency-aware multi-task dispatching and co-location in federated edge container-instances," in *2020 IEEE International Conference on Edge Computing (EDGE)*, 2020, pp. 91–98.

[10] C. Costache, O. Machidon *et al.*, "Software-defined networking of linux containers," in *2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference*, Chisinau, Moldova, 2014, pp. 1–4.

[11] D. M. Manias and A. Shami, "The need for advanced intelligence in nfv management and orchestration," *IEEE Network*, vol. 35, no. 1, pp. 365–371, January/February 2021.

[12] H. Hawilo, M. Jammal *et al.*, "Orchestrating network function virtualization platform: Migration or re-instantiation?" in *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, Turin, Italy, 2013, pp. 1–6.

[13] A. Barbalace, M. L. Karaoui *et al.*, "Edge computing: The case for heterogeneous-isa container migration," in *Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 2020, pp. 73–87.

[14] T. G. Rodrigues, K. Suto *et al.*, "Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, 2016.

[15] M. Alam, J. Rufino *et al.*, "Orchestration of microservices for iot using docker and edge computing," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 118–123, 2018.

[16] K. Kaur, S. Garg *et al.*, "Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 44–51, 2018.

[17] O. I. Abdullaziz, L. C. Wang *et al.*, "Enabling mobile service continuity across orchestrated edge networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1774–1787, 2020.

[18] O. Oleghe, "Container placement and migration in edge computing: Concept and scheduling models," *IEEE Access*, vol. 9, pp. 68 028–68 043, 2021.

[19] K. Govindaraj and A. Artemenko, "Container live migration for latency critical industrial applications on edge computing," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2018, pp. 83–90.

[20] F. Wang, J. Xu *et al.*, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2017.

[21] M. Alicherry and T. Lakshman, "Optimizing data access latencies in cloud systems by intelligent virtual machine placement," in *2013 IEEE 32nd International Conference on Computer Communications (INFOCOM)*, Turin, Italy, 2013, pp. 647–655.

[22] S. Vaucher, R. Pires *et al.*, "Sgx-aware container orchestration for heterogeneous clusters," *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 730–741, 2018.

[23] D. M. Manias *et al.*, "Machine learning for performance-aware virtual network function placement," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[24] D. M. Manias, H. Hawilo, M. Jammal, and A. Shami, "Depth-optimized delay-aware tree (do-dat) for virtual network function placement," *IEEE Networking Letters*, vol. 2, no. 3, pp. 149–153, Sept. 2020.

[25] Raritan, "Datacenters of the future: A shifting landscape from the core to the edge." [Online]. Available: `https://www.raritan.com/eu/landing/datacenters-of-the-future-whitepaper/thanks`

[26] R. Alsurdeh, R. N. Calheiros *et al.*, "Hybrid workflow scheduling on edge cloud computing systems," *IEEE Access*, vol. 9, pp. 134 783–134 799, 2021.

[27] J. W. Chinneck, "Binary and mixed-integer programming," 2016.

[28] R. M. Karp, "Reducibility among combinatorial problems," pp. 85–103, 1972.

[29] Q. Liu, C. F. Kwong *et al.*, "A fuzzy-clustering based approach for madm handover in 5g ultra-dense networks," *Springer's Wireless Networks*, pp. 1–14, 2019.

[30] D. Burstein, "Edge computing architecture impact on latency," 2020. [Online]. Available: https://stlpartners.com/edge-computing/edge-computing-architecture-impact-latency/

[31] R. A. Addad, D. L. C. Dutra *et al.*, "Fast service migration in 5g trends and scenarios," *IEEE Network*, 2020.

[32] M. Terneborg, J. K. Rönnberg *et al.*, "Application agnostic container migration and failover," *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pp. 565–572, 2021.

[33] H. M. Dur, "A container-based code offloading framework for mobile edge computing applications," 2021, middle East Technical University.

[34] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Edge-enabled v2x service placement for intelligent transportation systems," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1380–1392, 2021.

[35] C. Bravo and H. B"ackstr"om, "Edge computing deployment strategies: Whitepaper," *Ericsson.com. Ericsson, April*, vol. 13, 2020. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/white-papers/edge-computing-and-deployment-strategies-for-communication-service-providers

[36] H. Hawilo, M. Jammal, and A. Shami, "Network function virtualization-aware orchestrator for service function chaining placement in the cloud," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 643–655, 2019.

[37] Cisco, "Cisco annual internet report (2018–2023)," 3 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[38] I. Shaer, A. Haque, and A. Shami, "Multi-component v2x applications placement in edge computing environment," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[39] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud–mec collaborative task offloading scheme with service orchestration," *in IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5792–5805, 7 2020.

[40] M. Abu Sharkh, A. Ouda, and A. Shami, "A resource scheduling model for cloud computing data centers," in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2013, pp. 213–218.

[41] TierPoint, "Edge computing strategic guide," 6 2020. [Online]. Available: `https://www.tierpoint.com/edge-computing-strategic-guide/`

[42] M. Bouet and V. Conan, "Geo-partitioning of mec resources," A. for Computing Machinery, Ed. New York, NY, USA: Workshop on Mobile Edge Communications (MECOMM '17), 2017, pp. 43–48.

[43] X. Guan, X. Wan, J. Wang, X. Ma, and G. Bai, "Mobility aware partition of mec regions in wireless metropolitan area networks," in - *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS),*, I. Infocom, Ed. HI: Honolulu, 2018, pp. 1–2.

[44] D. A. Tran, T. T. Do, and T. Zhang, *A Stochastic Geo-Partitioning Problem for Mobile Edge Computing.* in IEEE Transactions on Emerging Topics in Computing, 2020.

[45] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang, "Selective offloading in mobile edge computing for the green internet of things," *IEEE Network*, vol. 32, no. 1, pp. 54–60, 2018.

[46] N. Wang, X. Wang, P. Palacharla, T. Ikeuchi, and W. Xie, "Mobility-aware 5g midhaul network design for optimizing edge computing resources," in *2019 Optical Fiber Communications Conference and Exhibition (OFC).* IEEE, 2019, pp. 1–3.

[47] W. You, C. Dong, X. Cheng, X. Zhu, Q. Wu, and G. Chen, *Joint Optimization of Area Coverage and Mobile Edge Computing with Clustering for FANETs.* in IEEE Internet of Things Journal, 2020.

[48] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge, and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, p. 2017, 2017.

[49] M. M. Lopes, W. A. Higashino, M. A. M. Capretz, and L. F. Bittencourt, "My-ifogsim: A simulator for virtual machine migration in fog computing," *In UCC ' Companion*, vol. 17, pp. 47–52, 2017.

[50] C. Sonmez, A. Ozgovde, and C. E. Ersoy, "An environment for performance evaluation of edge computing systems," *Trans Emerging Tel Tech*, vol. 29, 2018. [Online]. Available: `https://doi.org/10.1002/ett.3493`

[51] I. Lera, C. Guerrero, and C. Juiz, "Yafs: A simulator for iot scenarios in fog computing," *IEEE Access*, vol. 7, pp. 91 745–91 758, 2019.

[52] G. Etsi, "Multi-access edge computing (mec); framework and reference architecture," *ETSI, Tech*, vol. 2019, 2019. [Online]. Available: `https://www.etsi.org/deliver/etsi`

[53] F. Callegati, W. Cerroni, C. Contoli, and G. Santandrea, "Performance of network virtualization in cloud computing infrastructures: The openstack case," in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*. IEEE, 2014.

[54] A. Moubayed, M. Injadat, A. Shami, and H. Lutfiyya, "Dns typo-squatting domain detection: A data analytics & machine learning based approach," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–7.

[55] L. Yang and A. Shami, "Ids-ml: An open source code for intrusion detection system development using machine learning," *Software Impacts*, vol. 14, pp. 100–446, 2022.

[56] D. Melkov and S. Paulikas, "Security benefits and drawbacks of software-defined networking," in *IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, 2021.

[57] A. Kak, V.-Q. Pham, H.-T. Thieu, and N. Choi, "Demo: A disaggregated o-ran platform for network slice deployment and assurance," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–2.

[58] D. Rafique and L. Velasco, "Machine learning for network automation: overview, architecture, and applications [invited tutorial]," *Journal of Optical Communications and Networking*, 2018.

[59] A. Sebbar, K. Zkik, Y. Baddi, M. Boulmalf, and M. D. Ech-Cherif El Kettani, "Secure data sharing framework based on supervised machine learning detection system for future sdn-based networks," in *Machine Intelligence and Big Data Analytics for Cybersecurity Applications*. Springer, 2021.

[60] T. Tetcos, "Tetcos: Netsim - network simulation software, india," 2022. [Online]. Available: `https://www.tetcos.com/`

[61] A. Oliveira and T. Vazão, "Generating synthetic datasets for mobile wireless networks with sumo," in *Proceedings of the 19th ACM International Symposium on Mobility Management and Wireless Access*, 2021.

[62] p. networks, "Nettest 5g network emulators," 2022. [Online]. Available: `https://www.polarisnetworks.net/5g-network-emulators.html`

[63] A. Oliveira and T. Vazão, "Generating synthetic datasets for mobile wireless networks with sumo," 2021.

[64] M. Kalil, A. Shami, A. Al-Dweik, and S. Muhaidat, "Low-complexity power-efficient schedulers for lte uplink with delay-sensitive traffic," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4551–4564, Oct. 2015.

[65] S. R. Pokhrel, J. Ding, J. Park, O.-S. Park, and J. Choi, "Towards enabling critical mmtc: A review of urllc within mmtc," *IEEE Access*, vol. 8, pp. 131 796–131 813, 2020.

[66] E. Aqeeli, A. Moubayed, and A. Shami, "Power-aware optimized rrh to bbu allocation in c-ran," *IEEE Transactions on Wireless Communications*, vol. 17, no. 2, pp. 1311–1322, 2017.

[67] M. Jinno, H. Takara, B. Kozicki, Y. Tsukishima, Y. Sone, and S. Matsuoka, "Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies," *IEEE Communications Magazine*, vol. 47, no. 11, 2009.

[68] l. qiang, "Fixed grid vs flexible grid," *Huawei Enterprise Support Community*, 2020. [Online]. Available: `https://forum.huawei.com/enterprise/en/fixed-grid-vs-flexible-grid/thread/641227-875`

[69] F. S. Souza, D. L. Guidoni, and G. R. Mateus, "Formulations for the rwa problem with traffic grooming, protection and qos in wdm optical networks," *Proceedings - IEEE Symposium on Computers and Communications*, 2012.

[70] O. Awwad, A. I. Al-Fuqaha, and A. Rayes, "Traffic grooming, routing, and wavelength assignment in wdm transport networks with sparse grooming resources," *Computer Communications*, vol. 30, no. 18, pp. 3508–3524, 2007.

[71] L.-H. Lee, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo, and P. Hui, "All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda," *arXiv preprint*, 2021.

[72] B. C. Chatterjee, S. Ba, and E. Oki, "Fragmentation problems and management approaches in elastic optical networks: A survey," *IEEE Communications Surveys and Tutorials*, vol. 20, 2018.

[73] A. Celik, A. AlGhadhban, B. Shihada, and M.-S. Alouini, "Design and provision of traffic grooming for optical wireless data center networks," *IEEE Transactions on Communications*, vol. 67, no. 3, 2019.

[74] I. Corporation, "Evolving the awareness of optical networks," *The Evolution of Optical Networking*, 2019. [Online]. Available: `https://www.infinera.com/white-paper/evolving-the-awareness-of-optical-networks/`

[75] L. Yang and A. Shami, "A lightweight concept drift detection and adaptation framework for iot data streams," *IEEE Internet of Things Magazine*, vol. 4, no. 2, pp. 96–101, 2021.

[76] S. Aleyadeh, A. Moubayed, P. Heidari, and A. Shami, "Optimal container migration/re-instantiation in hybrid computing environments," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 15–30, 2022.

[77] 5G Americas, "Global 5G Connections Forcast," Accessed Oct. 26, 2022, 2022. [Online]. Available: `https://www.5gamericas.org/wp-content/uploads/2022/03/Global-5G-forecast.png`

[78] H. Hawilo, M. Jammal, and A. Shami, "Network function virtualization-aware orchestrator for service function chaining placement in the cloud," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 643–655, 2019.

[79] X. Yu, L. Lu, Q. Zhu, Y. Zhao, A. Nag, and J. Zhang, "Spectrum-entropy-minimized routing and spectrum allocation in ip over mixed-fixed/flex-grid optical networks," *Photonics*, vol. 9, 2022.

[80] P. Papanikolaou, K. Christodoulopoulos, and E. Varvarigos, "Joint multilayer planning of survivable elastic optical networks," in *2016 Optical Fiber Communications Conference and Exhibition (OFC)*, 2016, pp. 1–3.

[81] B. Zhao, X. Chen, J. Zhu, and Z. Zhu, "Survivable control plane establishment with live control service backup and migration in sd-eons," *Journal of Optical Communications and Networking*, vol. 8, no. 6, 2016.

[82] W. Jin, R. Gu, Y. Tan, and Y. Ji, "Proactive grooming with delay optimization in sliceable elastic optical network," *IEEE Access*, vol. 7, 2019.

[83] H. Kaur and M. Rattan, "Wind driven based heuristic solution for multiobjective traffic grooming in optical networks," *Wireless Personal Communications*, vol. 110, no. 3, 2020.

[84] C. Lee, M. Noh, and W. Seok, "Fast convergence bio-inspired traffic grooming for energy-efficient ip-over-wdm networks," in *2020 International Conference on*

Information and Communication Technology Convergence (ICTC), 2020, pp. 1064–1066.

[85] D. M. Manias, M. Jammal, H. Hawilo, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Machine learning for performance-aware virtual network function placement," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[86] I. Khan, L. Tunesi, M. Chalony, E. Ghillino, M. U. Masood, J. Patel, P. Bardella, A. Carena, and V. Curri, "Machine-learning-aided abstraction of photonic integrated circuits in software-defined optical transport," in *Next-Generation Optical Communication: Components, Sub-Systems, and Systems X*, vol. 11713, 2021, pp. 145–150.

[87] H. Yang, Q. Yao, A. Yu, Y. Lee, and J. Zhang, "Resource assignment based on dynamic fuzzy clustering in elastic optical networks with multi-core fibers," *IEEE Transactions on Communications*, vol. 67, no. 5, 2019.

[88] Y. Zhao, B. Yan, D. Liu, Y. He, D. Wang, and J. Zhang, "Soon: self-optimizing optical networks with machine learning," *Optics express*, vol. 26, no. 22, 2018.

[89] F. Musumeci, C. Rottondi, A. Nag, I. Macaluso, D. Zibar, M. Ruffini, and M. Tornatore, "An overview on application of machine learning techniques in optical networks," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, 2018.

[90] B. Yan, Y. Zhao, Y. Li, X. Yu, J. Zhang, Y. Wang, L. Yan, and S. Rahman, "Actor-critic-based resource allocation for multi-modal optical networks," in *2018 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2018.

[91] X. Chen, J. Guo, Z. Zhu, R. Proietti, A. Castro, and S. B. Yoo, "Deep-rmsa: A deep-reinforcement-learning routing, modulation and spectrum assignment agent for elastic optical networks," *Optica Publishing Group*, 2018.

[92] R. Proietti, X. Chen, K. Zhang, G. Liu, M. Shamsabardeh, A. Castro, L. Velasco, Z. Zhu, and S. J. Ben Yoo, "Experimental demonstration of machine-learning-aided qot estimation in multi-domain elastic optical networks with alien wavelengths," *Journal of Optical Communications and Networking*, vol. 11, no. 1, 2019.

[93] M. A. Sharkh, A. Kanso, A. Shami, and P. Öhlén, "Building a cloud on earth: A study of cloud computing data center simulators," *Computer Networks*, vol. 108, pp. 78–96, 2016.

[94] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2018.

[95] K. Koutlia, B. Bojovic, Z. Ali, and S. Lagén, "Calibration of the 5g-lena system level simulator in 3gpp reference scenarios," *Simulation Modelling Practice and Theory*, vol. 119, 2022.

[96] Y. Zhang, Y. Zhang, S. K. Bose, and G. Shen, "Migration from fixed to flexible grid optical networks with sub-band virtual concatenation," *Journal of Lightwave Technology*, vol. 35, no. 10, 2017.

[97] J. Santos, A. Eira, and J. Pires, "A heuristic algorithm for designing otn over flexible-grid dwdm networks." *J. Commun.*, vol. 12, no. 9, 2017.

[98] S. Aleyadeh, A. Moubayed, P. Heidari, and A. Shami, "Optimal container migration/re-instantiation in hybrid computing environments," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 15–30, 2022.

[99] A. Chouman, D. M. Manias, and A. Shami, "A reliable amf scaling and load balancing framework for 5g core networks," in *2023 International Wireless Communications and Mobile Computing (IWCMC)*, 2023, pp. 252–257.

[100] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *arXiv preprint arXiv:2202.01032*, 2022.

[101] H. Lee, J. Cha, D. Kwon, M. Jeong, and I. Park, "Hosting AI/ML workflows on O-RAN RIC platform," in *IEEE Globecom Workshops (GC Wkshps.* IEEE, 2020, pp. 1–6.

[102] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic Steering for Service Function Chaining," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1, pp. 487–507, 2019.

[103] M. Dryjański, Ł. Kułacz, and A. Kliks, "Toward Modular and Flexible Open RAN Implementations in 6G Networks: Traffic Steering Use Case and O-RAN xApps," *Sensors*, vol. 21, no. 24, p. 8173, 2021.

[104] S. Wong, B. Han, and H. D. Schotten, "5g network slice isolation," *Network*, vol. 2, no. 1, pp. 153–167, 2022. [Online]. Available: https://www.mdpi.com/2673-8732/2/1/11

[105] F. Guillemin, A. Aimi, T. Kerdoncuff, and S. Rovedakis, "Reference architecture for slicing in lorawan networks (updated version)," Ph.D. dissertation, Orange Labs, 2022.

[106] R. Yu, G. Xue, and X. Zhang, "QoS-Aware and Reliable Traffic Steering for Service Function Chaining in Mobile Networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2522–2531, 2017.

[107] M. Karbalaee Motalleb, V. Shah-Mansouri, and S. Nouri Naghadeh, "Joint power allocation and network slicing in an open ran system," *arXiv e-prints*, pp. arXiv–1911, 2019.

[108] P. Foroughi, P. Martins, P. Nivaggioli, and J.-L. Rougier, "Slice-aware open radio access network planning and dimensioning," in *2022 IEEE 96th Vehicular Technology Conference (VTC)*. IEEE, 2022, pp. 1–7.

[109] J. Thaliath, S. Niknam, S. Singh, R. Banerji, N. Saxena, H. S. Dhillon, J. H. Reed, A. K. Bashir, A. Bhat, and A. Roy, "Predictive closed-loop service automation in o-ran based network slicing," *IEEE Communications Standards Magazine*, vol. 6, no. 3, pp. 8–14, 2022.

[110] I. Tamim, A. Saci, M. Jammal, and A. Shami, "Downtime-aware o-ran vnf deployment strategy for optimized self-healing in the o-cloud," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[111] Q. H. Duong, I. Tamim, B. Jaumard, and A. Shami, "A column generation algorithm for dedicated-protection o-ran vnf deployment," in *2022 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2022, pp. 1206–1211.

[112] G. O. Boateng, G. Sun, D. A. Mensah, D. M. Doe, R. Ou, and G. Liu, "Consortium blockchain-based spectrum trading for network slicing in 5g ran: A multi-agent deep reinforcement learning approach," in *IEEE Transactions on Mobile Computing*, vol. 22, no. 10. IEEE, 2023, pp. 801–815.

[113] A. Filali, B. Nour, S. Cherkaoui, and A. Kobbane, "Communication and computation o-ran resource slicing for urllc services using deep reinforcement learning," *IEEE Communications Standards Magazine*, vol. 7, no. 1, pp. 66–73, 2023.

[114] H. Chergui, L. Blanco, L. A. Garrido, K. Ramantas, S. Kukliński, A. Ksentini, and C. Verikoukis, "Zero-touch ai-driven distributed management for energy-efficient 6g massive network slicing," *IEEE Network*, vol. 35, no. 6, pp. 43–49, 2021.

[115] M. Mirahmadi, A. Al-Dweik, and A. Shami, "Interference modeling and performance evaluation of heterogeneous cellular networks," *IEEE Transactions on Communications*, vol. 62, no. 6, pp. 2132–2144, 2014.

[116] A. Moubayed, A. Shami, and A. Al-Dulaimi, "On end-to-end intelligent automation of 6g networks," *Future Internet*, vol. 14, no. 6, p. 165, 2022.

[117] D. M. Manias, A. Chouman, and A. Shami, "Model drift in dynamic networks," *IEEE Communications Magazine*, vol. 61, no. 10, pp. 78–84, 2023.

[118] M. A. Siddiqi, H. Yu, and J. Joung, "5g Ultra-Reliable Low-Latency Communication Implementation Challenges and Operational Issues with IoT Devices," *Electronics*, vol. 8, no. 9, 2019. [Online]. Available: `https://www.mdpi.com/2079-9292/8/9/981`

Sam Aleyadeh

EDUCATION

University of Western Ontario London, Ontario
Ph.D. Computer Anthropology 2018 - Present
Dissertation: *Towards Zero Touch Next Generation End-to-End Network Management*
Academic advisor: *Dr. Abdallah Shami*

Queens University Kingston, Ontario
M.Sc. Computer Science 2012 - 2014
Dissertation: *Road And Driver Monitoring System*
Academic advisor: *Dr. Hossam Hassanein*

Queens University Kingston, Ontario
B.Sc. Computer Engineering 2007 - 2012

PUBLICATIONS

Published

Sam Aleyadeh, Abbas Javadtalab, and Abdallah Shami. Modular Simulation Environment Towards OTN AI-based Solutions. *arXiv preprint arXiv:2306.11135, 2023.*

Sam Aleyadeh, Abdallah Moubayed, Parisa Heidari, and Abdallah Shami. Optimal container migration/re-instantiation in hybrid computing environments. *IEEE Open Journal of the Communications Society, 2022.*

Sam Aleyadeh, Abdallah Moubayed, and Abdallah Shami. Mobility aware edge computing segmentation towards localized orchestration. *2021 International Symposium on Networks, Computers and Communications, 2021.*

Najah Abu Ali, **Sam Aleyadeh**, Fatiha Djebbar, Akram Alomainy, Maha Muhamed Ali Almaazmi, and Shaikha Al Ghaithi. Performance evaluation of routing protocols in electromagnetic nanonetworks. *IEEE Access, 2018.*

Sam Aleyadeh, and Abed El-Hamid Taha An IoT-Based architecture for waste management. *IEEE International Conference on Communications Workshops, 2018.*

Najah Abu Ali, **Sam Aleyadeh**, Fatiha Djebbar, and Akram Alomainy Evaluation of data dissemination schemes in electromagnetic nanosensor networks. *IEEE Global Communications Conference, 2018.*

Sam Aleyadeh, Sharief Oteafy, Hossam Hassanein Scalable Transportation Monitoring Using The Smartphone Road Monitoring (SRoM) System. *The 5th ACM symposium on the development and analysis of intelligent vehicular networks and applications, 2015.*

Under Review

Sam Aleyadeh, Abbas Javadtalab, and Abdallah Shami Throughput Latency Targeted RL Spectrum Allocation In Heterogeneous OTN. *journal*

Sam Aleyadeh, Ibrahim Tamim, and Abdallah Shami Transfer Learning Accelerated Network Slice Management for Next Generation Services *journal*

RESEARCH AND WORK EXPERIENCE

University of Western Ontario London, On

Research Assistant, Optical Transport Networking Spectrum Allocation project 2021-Present

Created an end-to-end modular simulator encompassing user mobility, 5G traffic generation, and optical transport and multiplexing. as well as developed a novel Spectrum allocation solution utilizing Reinforcement learning agents on isolated optical paths to improve throughput utilization through lower spectral fragmentation.

Academic advisor: Dr. Abdallah Shami

Research Assistant, Edge Computing container orchestration Project 2018-2021

Formulated a Mixed Integer Linear Programming orchestrator for containers within edge computing environments developed specifically for highly stringent services targeting lower downtimes. Extended the solution into a viable heuristic solution for realistic live deployments beyond benchmarking purposes only.

Academic advisor: Dr. Abdallah Shami

Alfaisal University Riyadh, Kingdom of Saudi Arabia

Primary Researcher, Smart City Waste Management System 2015-2016

Designed and implemented an Android and Arduino-based prototype-based smart city system for monitoring and organizing garbage collection cycles. The system controls collection routes utilizing a predictive algorithm to minimize impact on ongoing traffic as well as prevent overfilled garbage bins

Queens University Kingston, Ontario

Research Assistant, Transportation and Road Condition Monitoring Project 2013-2014

Designed and implemented an Android client-server system and prototype that uses smartphones and vehicle onboard diagnostic tools to monitor roads for potholes, in addition to dangerous driving patterns (high-speed weaving, hard stops, and drifting). Upon detection, road-related reports are sent to the map engine to populate interactive and heat maps, while driving-related reports are sent with relevant data to law enforcement agencies.

Academic advisor: Dr. Hossam Hassanein

Researcher, Wireless Body Area Network Project 2012-2013

Designed and implemented an Android system capable of monitoring vital health signs such as ECG and detecting health anomalies from the collected data. The prototype is built using shimmer sensors (WBAN), a smartphone, a WBAN smartphone interface, and a database server. The system allows healthcare providers to monitor live vitals from their smartphones; upon detection, the system alerts the incident's preconfigured emergency service facilities.