TU WIEN Informatics

Multi-Agent Systems in Vehicular Edge Computing

A Communication-Centric Approach to Task Handovers

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Nikolaus Spring, BSc
Matrikelnummer 11908527

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ. Prof. Dr. Schahram Dustdar
Mitwirkung: Univ. Ass. Dr. Andrea Morichetta

Wien, 27. August 2024

Nikolaus Spring	Schahram Dustdar

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Multi-Agent Systems in Vehicular Edge Computing

A Communication-Centric Approach to Task Handovers

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Nikolaus Spring, BSc
Registration Number 11908527

to the Faculty of Informatics

at the TU Wien

Advisor: Univ. Prof. Dr. Schahram Dustdar
Assistance: Univ. Ass. Dr. Andrea Morichetta

Vienna, August 27, 2024

Nikolaus Spring	Schahram Dustdar

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Nikolaus Spring, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 27. August 2024

Nikolaus Spring

Danksagung

Mein aufrichtiger Dank gilt in erster Linie meinen Betreuern, Univ. Prof. Dr. Schahram Dustdar und Dr. Andrea Morichetta, für ihre unschätzbare Unterstützung, ihre wertvollen Anregungen und ihr unermüdliches Engagement während meiner gesamten Forschungsarbeit. Ihre Expertise und Ermutigung haben meine Diplomarbeit erheblich bereichert und mich inspiriert, mein Wissen stetig zu erweitern.

Ebenso möchte ich mich bei der Firma Bosch herzlich bedanken, die mich während des gesamten Forschungsprozesses unterstützt hat. Von der gemeinsamen Entwicklung eines praxisrelevanten Themas bis hin zur kontinuierlichen Begleitung und wertvollen fachlichen Impulsen war ihre Unterstützung von großem Wert. Besonders danke ich Manuel Pascual Garcia-Tubio für die bereichernden Diskussionen und praxisnahen Einsichten, die die Tiefe und den Einfluss meiner Arbeit maßgeblich verstärkt haben.

Schließlich möchte ich meiner Familie meinen tiefsten Dank aussprechen, deren unerschütterliche Unterstützung das Fundament meiner Arbeit bildete. Ein besonderer Dank gilt meiner Freundin Carina, deren Verständnis, Geduld und stetige Ermutigung mir die Kraft und Motivation gegeben haben, diese herausfordernde, aber lohnende Erfahrung erfolgreich zu meistern.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Univ. Prof. Dr. Schahram Dustdar and Dr. Andrea Morichetta, for their invaluable guidance, insightful feedback, and unwavering support throughout this research journey. Their expertise and encouragement have not only enriched my work but also inspired me to push the boundaries of my knowledge.

I am sincerely thankful to Bosch for their continuous support throughout the entire process of my research, from helping me identify a topic with profound real-world relevance to providing ongoing assistance and valuable insights. I especially extend my heartfelt appreciation to Manuel Pascual Garcia-Tubio, whose thoughtful discussions and practical perspectives significantly enhanced the depth and impact of my work.

Finally, I would like to extend my deepest thanks to my family, whose constant support has been my foundation throughout this journey. A special thanks goes to my girlfriend, Carina, whose understanding, patience, and unwavering encouragement have been a source of strength and motivation, making this challenging yet rewarding experience possible.

Kurzfassung

Die rasante Entwicklung von Fahrzeugtechnologien, insbesondere im autonomen Fahren und bei Fahrerassistenzsystemen (ADAS), hat die Anforderungen an die Rechenleistung erheblich gesteigert und die Grenzen traditioneller, zentralisierter Cloud-Computing-Ansätze aufgezeigt. Eine der größten Herausforderungen ist die Latenz, die durch die physische Entfernung zwischen Fahrzeugen und entfernten Cloud-Servern entsteht und sich in zeitkritischen Anwendungen negativ auswirken kann. „Vehicular Edge Computing“ (VEC) bietet eine Lösung, indem Rechenressourcen näher an die Fahrzeuge gebracht werden und rechenintensive Aufgaben an nahegelegene, sogenannte Roadside Units (RSUs) ausgelagert werden. Allerdings stellen die Dynamik und Mobilität von Fahrzeugnetzen erhebliche Herausforderungen bei der Verwaltung von Aufgabenübergaben (engl. Handover) zwischen RSUs dar, wobei die Minimierung der Latenz und die Sicherstellung eines kontinuierlichen Dienstes entscheidend für die Systemleistung und -sicherheit sind.

Diese Arbeit stellt die „Agent-based RSU Handover Coordination“ (ARHC) Strategie vor, einen neuartigen Ansatz, der Multi-Agenten-Systeme (MAS) in RSUs integriert, um die Entscheidungsfindung zu dezentralisieren und Handover in VEC-Umgebungen zu optimieren. Die ARHC-Strategie verbessert die Netzwerkleistung, indem unnötige Handover reduziert, Rechenlasten effizient ausgeglichen und die Dienstgüte (engl. Quality of Service, QoS) gesteigert werden. Um die Wirksamkeit der ARHC-Strategie zu bewerten, wurde eine Simulationsumgebung entwickelt, die einen detaillierten Vergleich mit konventionellen Handover-Methoden ermöglichte. Diese Bewertung konzentrierte sich auf Latenz, Kommunikationseffizienz und Skalierbarkeit und lieferte eine umfassende Analyse der Auswirkungen.

Die Ergebnisse zeigen, dass die ARHC-Strategie die Anpassungsfähigkeit, Zuverlässigkeit und Effizienz von VEC-Systemen deutlich verbessert, insbesondere in Szenarien mit geringen Latenz- und hohen Zuverlässigkeitsanforderungen. Durch die Dezentralisierung der Kontrolle an den Netzwerkrand wird der Kommunikationsaufwand reduziert und die Ressourcennutzung optimiert.

Diese Forschung bietet ein solides Framework für das Management von Handover in VEC und stellt bedeutende Fortschritte in der Entwicklung und Implementierung von Intelligenten Verkehrssystemen dar.

Abstract

The rapid evolution of vehicular technologies, particularly in Autonomous Driving (AD) and Advanced Driver Assistance Systems (ADAS), has dramatically increased computational demands, revealing critical limitations in traditional centralized cloud computing models. One of the primary challenges is the latency caused by the physical distance between vehicles and remote cloud servers, which can be detrimental in time-sensitive applications. Vehicular Edge Computing (VEC) addresses this issue by bringing computational resources closer to the vehicles, enabling real-time data processing by offloading computationally intensive tasks to nearby Road Side Units (RSUs). However, vehicular networks' dynamics and mobile nature introduce significant challenges in managing task handovers between RSUs, where minimizing latency and ensuring continuous, uninterrupted service are crucial for maintaining system performance and safety.

This thesis introduces the Agent-based RSU Handover Coordination (ARHC) strategy, a novel approach that integrates Multi-Agent Systems (MASs) within RSUs to decentralize handover decision-making and optimize task handovers in VEC environments. The ARHC strategy moves away from traditional centralized or vehicle-based handover methods by shifting control to the network edge. This strategy enhances network performance by reducing unnecessary handovers, efficiently balancing computational loads, and improving overall Quality of Service (QoS). A custom simulation environment was developed to rigorously assess the effectiveness of the ARHC strategy, simulating diverse real-world traffic scenarios and variable network conditions. This evaluation focused on key metrics such as latency, communication efficiency, and system scalability, providing a comprehensive analysis of the strategy's impact.

The results demonstrate that the ARHC strategy significantly improves the adaptability, reliability, and efficiency of VEC systems, particularly in scenarios that demand low latency and high reliability. By decentralizing control to the network edge, the ARHC strategy effectively reduces communication overhead and optimizes resource utilization.

This research offers a robust framework for task handover management in VEC, delivering significant advancements in the development and implementation of Intelligent Transportation Systems (ITS).

Contents

CHAPTER 1

Introduction

1.1 Background

The rapid evolution of vehicular technology, particularly in Autonomous Driving (AD) and Advanced Driver Assistance Systems (ADAS), has led to increasingly complex computational demands. These applications require real-time data processing, low-latency communication, and high-bandwidth capabilities, requirements that surpass the capabilities of traditional centralized cloud computing due to latency introduced by the physical distance between vehicles and remote servers. In safety-critical applications like AD, where split-second decisions are essential, the delays caused by transmitting data to distant cloud servers are unacceptable. [13, 33, 28]

To address these challenges, Edge Computing (EC) has emerged as a promising solution by bringing computational resources closer to the vehicles themselves. By shifting computation to the "edge" of the network, closer to where the data is generated, EC significantly reduces latency, enabling real-time data processing and enhancing overall system responsiveness. [8, 21] In the context of vehicular networks, this approach is known as Vehicular Edge Computing (VEC), where computationally intensive tasks are offloaded from vehicles to nearby Road Side Units (RSUs) equipped with Multi-Access Edge Computing (MEC) servers. This offloading improves processing efficiency and alleviates the computational burden on individual vehicles, allowing them to focus on additional safety-critical tasks. [12, 25, 33]

However, the dynamic and mobile nature of vehicular environments introduces new challenges. As vehicles move, they frequently transition between network zones, necessitating seamless handover of tasks from one RSU to another to maintain service continuity. Whether centralized or vehicle-based, traditional handover strategies often struggle to scale and maintain consistent performance under these dynamic conditions. Efficient task handover is crucial for maintaining high Quality of Service (QoS), minimizing latency, and

ensuring uninterrupted data processing, especially as vehicles move out of range of their current RSU or as RSUs become overloaded [1, 29, 4]. In response to these challenges, advanced handover management strategies have become critical to achieving seamless connectivity and maintaining service quality in MEC-enabled vehicular networks. [4]

1.2 Problem Statement

While the concept of VEC holds great promise, the task handover process in such a dynamic environment remains a significant challenge. Traditional handover mechanisms, commonly used in cellular networks, are often inadequate for vehicular networks' high mobility and real-time demands. These conventional approaches, which rely on signal strength or fixed thresholds to trigger handovers, do not account for the complex, distributed nature of VEC systems, where computational load balancing and network congestion are critical factors. [34, 3]

To address these issues, this thesis explores the integration of Multi-Agent Systems (MASs) into the VEC framework. MASs, consisting of multiple autonomous agents working collaboratively, offer a decentralized approach to managing the handover process. By leveraging the inherent parallelism, flexibility, and robustness of MASs, the system can dynamically adapt to changing network conditions, optimize resource utilization, and improve the efficiency of task handovers. [5, 1]

1.3 Research Objectives and Contributions

The primary objective of this thesis is to develop and evaluate a novel strategy for managing task handovers in VEC environments. This strategy, termed Agent-based RSU Handover Coordination (ARHC), leverages the decentralization of decision-making to RSUs equipped with MEC capabilities. The ARHC strategy seeks to address the challenges posed by the dynamic and mobile nature of vehicular networks, optimizing handover processes to reduce unnecessary handovers, minimize communication overhead, and maintain high QoS across the network.[1]

This thesis makes the following significant contributions:

1. **Development of the ARHC Strategy**

   The ARHC strategy represents a novel approach to handover management in VEC environments by integrating MASs within RSUs. Unlike existing centralized or vehicle-based methods, this RSU-based approach enhances system reliability, scalability, and responsiveness in dynamic vehicular environments. By decentralizing the decision-making process, the ARHC strategy is designed to adapt to varying

[1] The source code and simulation environment implementing the ARHC strategy are available on GitHub: `https://github.com/nspring00/agent-based-rsu-handover-coordination-poc`.

network conditions, thereby improving the efficiency of task handovers and overall system performance.

2. **Creation of a Custom Simulation Environment**

   To thoroughly evaluate the ARHC strategy, this thesis develops a custom simulation environment explicitly tailored for VEC systems. This environment incorporates real-world traffic data, allowing for the realistic assessment of the proposed strategy's effectiveness in managing task handovers and optimizing resource allocation. The simulation environment also enables benchmarking of ARHC against traditional centralized and vehicle-based handover strategies, focusing on key metrics such as handover frequency, QoS, and load distribution.

This thesis aims to advance the state of the art in VEC by providing a robust framework for managing task handovers in dynamic, decentralized environments. By leveraging MASs and RSUs, the ARHC strategy has the potential to significantly improve the performance, scalability, and reliability of VEC systems.

1.4 Research Assumptions

The development and evaluation of the ARHC strategy within this thesis are based on several key assumptions that simplify the complex nature of VEC environments. These assumptions are critical to the scope and focus of this research, enabling a targeted analysis of handover coordination strategies.

1. **Network Model**: The network model assumes a structured environment with fixed RSUs locations. RSUs are assumed to have consistent, stable capacities and are interconnected via a reliable, low-latency wired network. This assumption allows for focused evaluation of the ARHC strategy without the added complexity of varying RSU placements or capacities.

2. **Communication Conditions**: Ideal communication conditions are assumed, with no packet loss, delays, or failures in data exchange between RSUs and vehicles. This simplifies the simulation and analysis, allowing the research to concentrate on the efficiency of the handover process itself rather than the potential complications introduced by communication issues.

3. **Computational Load Distribution**: Vehicles are assumed to offload a static portion of their computational load to nearby RSUs. This assumption focuses the study on managing these offloaded tasks by the RSUs, excluding the complexities of dynamic offloading decisions.

4. **Simulation Environment**: The simulation environment is designed to reflect typical urban settings with controlled conditions. While the primary focus is on normal operation scenarios, the impact of an RSU failure is also evaluated.

Although the ARHC strategy was not initially designed to handle such failures, this evaluation provides valuable insights into the robustness of the system and highlights areas for potential improvement.

These assumptions are elaborated in greater detail in Section 4 (Approach) and Section 6.4 (Discussion).

1.5 Structure of this Thesis

This thesis is organized as follows:

- **Chapter 1: Introduction**
  This chapter provides an overview of the background, outlines the problem statement and research objectives, and discusses the research assumptions, including the novelty and contributions of this thesis. It sets the foundation for the chapters that follow.
- **Chapter 2: Preliminaries**
  Introduces key concepts such as EC, VEC, and MASs, providing the necessary background for the research. It establishes the foundational concepts and technologies necessary for understanding the subsequent chapters.
- **Chapter 3: Related Work**
  This chapter reviews existing literature on handover coordination in VEC and related fields, identifying the limitations of current approaches and highlighting the need for the ARHC strategy.
- **Chapter 4: Approach**
  This chapter details the development of the ARHC strategy, including the motivation for adopting a decentralized approach, the network model, and the agent architecture. It also introduces the custom simulation environment to evaluate ARHC.
- **Chapter 5: Implementation**
  This chapter presents the implementation of the ARHC strategy within the simulation environment and the simulation environment itself.
- **Chapter 6: Evaluation**
  This chapter analyzes the performance of the ARHC strategy compared to traditional handover methods using the metrics outlined in the research objectives.
- **Chapter 7: Conclusion and Future Work**
  The final chapter summarizes the findings of the thesis, discusses the research implications, and suggests directions for future work.

CHAPTER 2

Preliminaries

This chapter provides essential background for the thesis. We begin with an overview of EC (Section 2.1), focusing on MEC (Section 2.2) and VEC (Section 2.3). These sections will cover the fundamental concepts, technologies, and EC applications, emphasizing mobile and vehicular contexts. For a concise comparison of these paradigms, refer to Table 2.1 at the end of Section 2.3. Finally, we discuss MASs (Section 2.4), including their architecture, applications, and associated challenges.

2.1 Edge Computing (EC)

2.1.1 Introduction to Edge Computing

Edge Computing (EC) extends cloud computing services closer to the end user, addressing high latency and mobility challenges inherent in traditional cloud models. By positioning computational resources at the network's edge, EC enhances application responsiveness and real-time data processing by improving speed and efficiency. This is particularly beneficial with the Internet of Everything (IoE), where the proliferation of smart devices generates vast amounts of data that require efficient local processing. [8, 21]

Key characteristics of EC include dense geographical distribution, mobility support, location awareness, and proximity to users, contributing to low latency and context awareness. [21] Local data processing at the edge enhances security, reduces bandwidth usage, and improves energy efficiency. [8]

EC architecture typically involves four layers: the device layer, the edge layer, the fog layer, and the cloud layer. The device (or terminal) layer includes all connected devices, while the edge layer consists of nodes like base stations and edge servers that provide localized processing. The fog layer acts as an intermediary, extending the capabilities of the edge by providing additional computational resources and storage and enabling more complex

processing closer to the data source. The cloud layer handles large-scale data analysis and long-term storage, providing global insights and centralized management [8, 21].

Security and privacy are crucial in EC, involving encryption, secure data sharing, identity authentication, and privacy protection measures. The ability to employ fine-grained access control further enhances security. [8]

EC applications include video caching, 5G communication, network video live broadcasting, predictive maintenance, and security monitoring. Integration with 5G networks leverages EC for faster processing and low latency, enhancing these networks' capabilities. [8, 21]

2.1.2 General Network Setup

In EC, the network setup strategically deploys computational resources and networking capabilities to improve performance, security, and scalability. This setup is crucial for ensuring efficient data processing, low latency, and an enhanced user experience, particularly in situations that demand quick response times and localized data handling. [4, 8, 21]

Key Components

- **Base Stations**: Base stations are essential components of EC architectures, serving as the primary contact points for end-user devices and managing local data processing and transmission. In mobile networks, these stations are equipped with computational capabilities, traditionally found in centralized cloud servers, which help to decrease latency and enhance application responsiveness. [4, 8, 21]

- **Edge Servers**: Distributed across numerous locations, edge servers are strategically positioned closer to end-users than traditional cloud data centers. They manage data storage, processing, and real-time analytics, effectively minimizing the data transmitted to the cloud, thereby lowering latency and bandwidth usage. These servers can be in cellular base stations, Wi-Fi routers, or dedicated edge data centers. [8, 21]

- **Gateways**: Gateways serve as intermediaries between end-user devices and edge servers. They aggregate data from multiple sources, perform initial data filtering and preprocessing, and then forward relevant information for further processing. Gateways are essential for managing data traffic and ensuring efficient communication within the network. [4, 8, 21]

- **Access Points (APs)**: Wi-Fi routers and similar devices function as access points, enabling connectivity for end-user devices and linking local networks with the wider EC infrastructure. Additionally, they can process and cache data to improve performance and minimize latency. [21]

Deployment Strategies

- **Proximity-Based Deployment:** The deployment of edge servers and base stations near end-users has been shown to minimize latency and enhance data processing speeds. This particularly benefits urban areas with high data traffic and user density. [21]
- **Regional Data Centers:** The establishment of regional data centers with edge servers facilitates geographically distributed data processing, effectively balancing the load across the network and reducing reliance on centralized cloud data centers. [21]
- **Hybrid Deployment:** A hybrid deployment model that integrates edge and cloud resources allows for flexible data processing and storage. Mission-critical and latency-sensitive applications can be handled at the edge, while less time-sensitive tasks can be processed in the cloud. [21]

2.2 Multi-Access Edge Computing (MEC)

Multi-Access Edge Computing (MEC), or Mobile Edge Computing, is a specialized form of EC tailored for mobile environments and specifically addresses the unique challenges of mobile networks, such as high mobility and frequent handovers. [28]

2.2.1 Distinction Between General EC and MEC

While general EC reduces latency and improves bandwidth efficiency by processing data near its source, MEC extends these benefits to highly mobile scenarios by integrating cloud capabilities within the network's edge infrastructure. This integration allows for low-latency, high-bandwidth services to be delivered directly from base stations and other edge nodes, which is critical for applications requiring rapid data processing and minimal latency, such as Augmented Reality (AR), Virtual Reality (VR), and AD. [28, 29]

A key distinction of MEC is its inherent support for mobility. Mobile devices frequently change their point of attachment to the network, necessitating robust handover mechanisms that ensure service continuity without performance degradation. This involves maintaining network connectivity and guaranteeing the seamless transfer of application states and data between edge nodes. [4]

2.2.2 Traditional Handover in Cellular Networks

Handover is a fundamental mechanism in cellular networks, facilitating the transfer of an ongoing call or data session from one cell to another as a user moves through the network. This process is critical for maintaining service continuity and quality. Traditional handover mechanisms rely on several triggers and methods to optimize performance and reduce disruptions. [34]

The most straightforward method uses the Received Signal Strength (RSS) of the current and candidate base stations. The handover occurs when the signal from the new base station becomes stronger than the current one, although this can lead to frequent and unnecessary handovers if the current signal is still adequate. To improve this, methods such as RSS with thresholds and hysteresis are employed. Thresholds ensure that the current signal must drop below a certain level before a handover is initiated, while hysteresis margins prevent the ping-pong effect, where a mobile repeatedly switches between two cells. Advanced techniques also include prediction-based methods, which forecast future signal strength to initiate handovers before the signal deteriorates significantly. [34]

Timers, such as dwell timers, are introduced to manage handover frequency, ensuring that a mobile device must stay connected to a new base station for a minimum period before another handover can occur. Dynamically adjusting hysteresis margins based on network conditions can further reduce unnecessary handovers and improve overall system performance. [34]

Handover Performance Metrics

Performance metrics for handover include call blocking probability, handover blocking probability, handover probability, call dropping probability, rate of handover, duration of interruption, and delay. These metrics are essential for evaluating the efficiency and reliability of handover processes in cellular networks. [34]

Soft vs Hard Handover

In terms of handover types, Hard Handover (HHO) involves breaking the old connection before the new one is activated, leading to brief service interruptions. In contrast, Soft Handover (SHO) establishes the new connection before releasing the old one, providing a seamless transition with no service interruption. [34]

In summary, understanding traditional handover mechanisms and their triggers is crucial for developing advanced handover strategies in MEC. These conventional methods lay the groundwork for more sophisticated approaches needed to handle the high mobility and low latency requirements of modern networks. [34]

2.2.3 Challenges and Solutions

Managing the handover process is a core challenge in MEC. Mobile devices move between different network cells, requiring a seamless transition of active sessions and services from one MEC server to another. This handover process must be efficient to avoid service interruptions and maintain QoS. [29]

Imperative vs. Alternative Handovers

Imperative handovers are critical for maintaining system stability and QoS. They are executed to prevent the mobile device from losing connectivity (when leaving the MEC

station's range) or to avoid the MEC station from being overloaded. In contrast, alternative handovers are not strictly necessary but can enhance system performance by balancing loads more effectively across MECs station. These handovers are initiated based on the potential to improve overall system efficiency and resource utilization.

Handover Strategies

1. **Handover Protocols**: MEC utilizes specialized handover protocols that differ from traditional mobile handovers. These protocols ensure that not just the network connection but also the application state and user context are smoothly transferred. [4, 23]

2. **Stateful and Stateless Migration**: MEC supports both stateful and stateless migration of services. Stateful migration involves transferring the entire application state, including memory and processing status, to the new MEC node. This allows the new node to resume precisely where the old one left off, thereby providing a seamless user experience. Stateless migration moves only the computation tasks, with session states reconstructed at the new node. [29, 4]

3. **Virtual Machine and Container Migration**: Techniques such as Virtual Machine (VM) and container migration are employed to move application instances between MEC nodes. Live migration techniques minimize downtime and service disruption. [29, 4]

Resource Allocation and Management

Effective resource management is crucial in MEC. Nodes must dynamically allocate resources to handle varying workloads while ensuring optimal performance and low latency:

1. **Dynamic Resource Allocation**: Algorithms for dynamic resource allocation help MEC nodes efficiently manage computing, storage, and network resources. These algorithms consider factors such as current load, predicted future demand, and QoS requirements to make real-time decisions. [4, 28, 29]

2. **Load Balancing**: Effective load balancing strategies are crucial to prevent any single MEC node from becoming a bottleneck. This involves distributing the workload evenly across multiple nodes to ensure high availability and reliability. [4, 29]

2.2.4 Specific Techniques and Processes

Handover Processes

The handover process in MEC involves several steps to ensure a seamless service transition:

1. **Network Discovery**: Before initiating a handover, the MEC system performs network discovery to identify potential target nodes that can accommodate the migrating service. [4]

2. **Decision Phase**: Once potential targets are identified, the system decides the optimal target based on criteria such as current load, latency, and resource availability. [4, 29]

3. **Execution Phase**: The actual handover involves migrating the application state, re-routing the network traffic, and updating the user context at the new MEC node. Techniques such as live migration ensure that this process happens with minimal disruption. [4, 29]

2.2.5 Handover in LTE vs. MEC

In traditional cellular networks like LTE networks, handover procedures are critical for maintaining ongoing connections and ensuring service continuity, as described in Section 2.2.2. The handover process in LTE typically involves three phases: preparation, execution, and completion: [23]

1. **Preparation Phase**: This phase involves the source LTE node selecting a target node based on measurement reports from the User Equipment (UE). The source node sends a handover request to the target node, which then performs admission control and allocates resources for the UE. [23]

2. **Execution Phase**: The UE disconnects from the source node and connects to the target node. Once connected, the UE sends a handover confirmation message to the target node to start receiving data. [23]

3. **Completion Phase**: This phase involves switching the downlink path to inform the core network entities to switch the data path from the source node to the target node and release the resources allocated to the UE in the source cell. [23]

In summary, MEC distinguishes itself from general EC through its robust support for mobility and seamless handover processes. By addressing the challenges of mobile environments, MEC enables a wide range of applications requiring low latency and high reliability, such as AD and real-time video analytics. Effective resource management and advanced migration techniques ensure these services remain uninterrupted as devices move across the network.

2.3 Vehicular Edge Computing (VEC)

Vehicular Edge Computing (VEC) is a specialized form of EC that integrates MEC capabilities within vehicular networks. By design, VEC addresses the high mobility,

real-time data processing needs, and significant computational requirements of modern vehicular applications, such as AD, AR, VR, and online gaming. [17, 25]

2.3.1 Key characteristics and components of VEC

VEC systems are characterized by their ability to handle dynamic and highly mobile environments. The key components of VEC include the RSU, the vehicular On-board Unit (OBU), and MEC servers.

RSUs and OBUs: RSUs are deployed along the roads and serve as the primary communication points for vehicles. They facilitate Vehicle-to-Infrastructure (V2I) communication. OBUs, installed in vehicles, handle data collection and preliminary processing. [17, 25]

MEC Servers: These servers are typically colocated with RSUs and provide significant computational resources for processing tasks offloaded from vehicles. This colocation minimizes the distance data needs to travel, thereby reducing latency and energy consumption. [17, 25]

2.3.2 Benefits of VEC

The integration of EC into vehicular networks offers multiple benefits:

- **Reduced Latency**: By processing data at the edge of the network, VEC significantly reduces the time required for data transmission, making it ideal for real-time applications. [17, 25]
- **Energy Efficiency**: Offloading computation-intensive tasks to nearby MEC servers reduces the energy consumption of vehicular OBUs, thereby extending the operational time of these units. [17, 25]
- **Enhanced Data Processing**: VEC systems can handle large volumes of data generated by vehicular sensors and applications, providing faster and more reliable data processing. [17, 25]

2.3.3 Challenges in VEC

Despite its benefits, VEC faces several challenges:

- **Mobility Management:** High mobility, characterized by the rapid and continuous movement of vehicles, necessitates robust handover mechanisms to maintain stable connectivity between RSUs and MEC servers. [17, 25]
- **Task Offloading Decisions:** Making optimal offloading decisions is crucial for minimizing energy consumption and ensuring low packet drop rates. Dynamic task offloading schemes are essential for adapting to varying network conditions and vehicle mobility. [17, 25]

Table 2.1: Differentiation of EC, MEC, and VEC

Feature	EC	MEC	VEC
Focus	General edge processing near the end-user.	EC tailored for mobile networks.	EC specialized for vehicular networks.
Mobility	Limited mobility support.	High mobility support with seamless handovers.	Very high mobility with rapid V2I handovers.
Key Applications	IoT, smart cities, local data processing.	Mobile services, AR/VR, low-latency mobile applications.	AD, V2X communication, real-time vehicle data processing.
Infrastructure	Edge servers, gateways, and APs.	Edge servers integrated with mobile base stations.	RSUs, OBUs, MEC servers.
Challenges	Resource management, security, latency reduction.	Seamless mobility, dynamic resource allocation.	High mobility management, low-latency processing under vehicular dynamics.

- **Resource Allocation:** Efficient allocation of computational resources among multiple vehicles is necessary to avoid congestion and ensure high QoS. Algorithms for dynamic resource allocation help manage the varying workloads and maintain system stability. [17, 25]

2.3.4 VEC in practice

In practical implementations, VEC systems use a combination of communication technologies such as Dedicated Short-Range Communication (DSRC), Wi-Fi, and cellular networks for V2I communication. MEC servers colocated with RSUs handle the offloaded tasks from vehicles, providing a powerful infrastructure for computation-intensive applications while reducing execution time and local energy consumption. [17, 25]

The deployment strategies in VEC involve placing MEC servers and RSUs in strategic locations to ensure optimal coverage and minimal latency. This setup is particularly beneficial in urban areas with dense traffic, with high demand for real-time data processing. [17, 25]

As shown in Table 2.1, there are key distinctions between EC, MEC, and VEC in terms of focus, mobility, key applications, infrastructure, and challenges.

2.4 Multi-Agent Systems (MAS)

Multi-Agent Systems (MASs) are a subset of Distributed Artificial Intelligence (DAI) where multiple agents interact within a common environment to achieve specific goals. These systems have gained traction due to their ability to handle complex, distributed, and dynamic problems more efficiently than traditional systems. The fundamental concept involves agents working collaboratively or competitively to solve problems beyond the capabilities of individual agents. [5, 32]

This structured overview provides a foundational understanding of MASs, setting the stage for exploring their integration into VEC for task handover coordination. The detailed examination of agents, communication, and coordination strategies will be critical for developing effective MAS solutions in this domain.

2.4.1 Agents in MASs

Balaji et al. [5] define an agent in MASs as a flexible, autonomous entity capable of perceiving its environment through sensors and acting upon it using actuators. According to them, agents possess several key properties:

- **Situatedness**: Interaction with the environment, encompassing both input and output, is direct and integral to the agent's design.
- **Autonomy**: Agents operate independently without external intervention, thereby isolating them from instability.
- **Inferential capability**: Agents can deduce new information from available data, enabling them to achieve abstract and complex objectives.
- **Responsiveness**: Agents can quickly adapt to changes in their environment.
- **Proactiveness**: Agents not only respond to changes but also take initiative to achieve their goals.
- **Social behavior**: Agents can communicate, collaborate, and share knowledge with other agents to accomplish tasks.

2.4.2 Characteristics of MASs

A Multi-Agent System (MAS) extends agent technology by incorporating a collection of loosely interconnected autonomous agents operating within an environment to pursue a shared objective. These agents may cooperate or compete, and they may choose to share or withhold knowledge from one another. [5, 37]

These systems are characterized by **decentralization**, meaning there is no central control, which enhances robustness and scalability. **Parallelism** is another key characteristic where agents operate concurrently, boosting system performance. Additionally,

MASs exhibit **flexibility and adaptability**, dynamically adjusting to new tasks and environments. [5, 37]

2.4.3 Challenges of MASs

Despite their advantages, MASs face several challenges. **Coordination complexity** is a primary issue, as ensuring effective coordination among agents to prevent conflicts and inefficiencies can be difficult. **Communication overhead** is another challenge involving the management of the volume of inter-agent communication to avoid performance bottlenecks. **Scalability issues** arise as the number of agents increases, and maintaining system performance under these conditions can be challenging. Moreover, **robustness and fault tolerance** are critical; MASs must be designed to handle agent failures gracefully without affecting overall system performance. [5, 37, 32]

2.4.4 Classifications of MASs

MASs can be classified based on their organizational structure:

- **Hierarchical Organization**: Agents are organized in a hierarchy with clear roles and responsibilities. [5]
- **Flat Organization**: All agents have equal roles and responsibilities. [5]
- **Hybrid Organization**: Combines elements of both hierarchical and flat structures to leverage their advantages. [5]

In addition, we also differentiate between homogeneous and heterogeneous structures. In **homogeneous structures**, agents share identical internal setups (goals, sensors, states, inference mechanisms, and actions) and differ only in their physical location and the environment they interact with. This uniformity is typically seen in distributed environments, where sensor input overlap is rare. Conversely, **heterogeneous structures** consist of agents with different abilities, structures, and functions, acting based on their specific environment and location. This diversity makes them more realistic and may result in conflicting goals. [5]

2.4.5 Performance metrics of MASs

Performance in MASs can be evaluated using metrics such as response time, communication efficiency, and system robustness. These metrics help assess how well the MAS achieves its goals under various conditions. For example, response time is a critical metric in assessing the QoS in MASs, particularly in systems with intensive communications like crisis management and cinema ticket selling scenarios. [5, 15]

2.4.6 Decision-making in MASs

Agents in MASs use various decision-making strategies to determine their actions. These strategies include:

- **Reactive Decision-Making**: Agents respond to changes in their environment in real-time. [5]
- **Deliberative Decision-Making**: Agents plan their actions based on predictions about the future state of the environment. [5]
- **Hybrid Decision-Making**: Combines reactive and deliberative approaches to balance immediate responses and long-term planning. [5]

2.4.7 Coordination in MASs

- **Protocols**: Sets of rules that define interactions between agents, such as the Contract Net Protocol, which manages task allocation among agents. [37]
- **Organizational Structures**: Defines roles and relationships between agents, ensuring structured and efficient interactions. [5]
- **Negotiation and Auctions**: Agents negotiate or bid for tasks to optimize resource allocation and task distribution. [5, 37]

2.4.8 Communication Patterns for MASs

Communication patterns ensure efficient and reliable data exchange among interconnected devices and systems within MASs. These patterns, which include request-response, publish-subscribe, discovery, message passing, and blackboard, provide solutions tailored to the unique requirements of MASs, allowing for better system interaction and coordination. [9, 11, 16]

Request-Response

In MASs, the request-response pattern is used for direct interactions where one agent (requester) sends a request to another agent (responder) and awaits a response. This pattern is often employed in task delegation and service provision among agents. Unlike traditional systems, MAS agents can dynamically switch roles between requester and responder, enhancing flexibility. However, the challenge lies in managing these dynamic roles and ensuring timely responses to maintain overall system performance. The pattern is handy for scenarios requiring immediate and deterministic responses but may introduce bottlenecks if many agents simultaneously request responses from a single agent. [9, 11, 16, 38]

Publish-Subscribe

The publish-subscribe pattern in MASs supports asynchronous communication, where agents publish messages to specific topics, and other agents subscribe to these topics to receive updates. This decoupling allows for flexible and scalable communication, making it effective in large-scale MASs. Agents use the publish-subscribe pattern to disseminate information about environmental changes, state updates, or resource availability. Acting as a neutral mediator, the event service propagates events to all relevant subscribers without needing publishers and subscribers to know each other. This decoupling in space, time, and synchronization makes the pattern highly flexible and scalable. [9, 11, 31]

Discovery

Discovery patterns enable agents to locate and connect dynamically without prior knowledge of network addresses. This capability is essential in environments where agents frequently join or leave the system. MASs often implement sophisticated discovery algorithms that allow agents to identify potential collaborators or competitors based on their goals and current states. Discovery services support dynamic, real-time interactions and adapt to the constantly changing agent configurations, enhancing the system's adaptability and responsiveness. [9, 16, 38]

Message Passing

Message passing in MASs is vital for direct agent-to-agent communication, supporting synchronous and asynchronous interactions. This pattern ensures real-time coordination and cooperation among agents. Optimized message-passing protocols handle high volumes of messages efficiently, maintaining low latency and reliability. Agents use this method to share state information, coordinate actions, and negotiate resources. Effective message-passing designs prevent communication bottlenecks and prioritize critical messages for timely delivery, ensuring the smooth operation of the MAS. [5, 38] Fault tolerance and reliability can be enhanced by incorporating redundancy in communication paths and error correction techniques, ensuring robust message delivery despite failures. [15]

Blackboard

The blackboard pattern in MASs acts as a shared knowledge repository where agents post and retrieve information. This pattern is particularly effective for collaborative problem-solving, requiring collective intelligence and coordination among multiple agents. Enhanced blackboard systems handle concurrency and prioritize critical information, allowing agents to share partial solutions, update task statuses, and coordinate activities, without direct communication. This indirect communication method simplifies managing complexity and ensures all agents access the latest necessary information for decision-making. [9, 37] Advanced techniques, such as Machine Learning (ML), can optimize the blackboard's efficiency, predicting which information will be needed and preemptively organizing it. [5]

CHAPTER 3

Related Work

This chapter reviews the existing literature on integrating MASs into VEC, focusing on communication strategies for task handover coordination. It comprehensively analyzes computation offloading, handover coordination, MAS integration, and security within VEC frameworks.

3.1 Computation Offloading in VEC

This section explores various strategies and methodologies for computation offloading in VEC, including Reinforcement Learning (RL) approaches, optimization techniques, and task assignment methods. These approaches are evaluated based on their ability to address challenges such as latency, energy consumption, and resource allocation in dynamic vehicular environments.

3.1.1 Reinforcement Learning Approaches

RL is particularly advantageous in VEC due to its ability to handle dynamic and uncertain environments where optimal strategies must be learned over time. Wang et al. propose a decentralized computation offloading algorithm using multi-agent imitation learning framed within a game theory approach to minimize task completion time in pervasive edge computing. Devices make independent decisions based on partial observations, achieving Nash equilibrium and outperforming other algorithms in efficiency and speed. Their method uniquely addresses decentralized computation offloading, ensuring fairness and optimizing task completion time through a game-theoretic framework. [40]

Alam et al. propose a Multi-agent Deep Reinforcement Learning-based Hungarian Algorithm (MDRLHA) for dynamic task offloading in VEC. Their cooperative three-layer Vehicle-Assisted Mobile Edge Computing (VMEC) network uses moving and parked vehicles as fog nodes, optimizing latency, energy consumption, and cost. MDRLHA leverages

the Hungarian algorithm for efficient task allocation, showing superior performance in high-mobility environments. Their approach emphasizes cooperative interactions and dynamic task management in high-mobility scenarios. [2]

Lin et al. propose a computation offloading strategy for VEC using Deep Reinforcement Learning (DRL) to address joint optimization of offloading failure rate and energy consumption. Based on a Markov Decision Process (MDP), their model considers task dependencies, vehicle mobility, and diverse computing resources, optimizing using a Deep Q-Network (DQN) with Simulated Annealing. This strategy effectively reduces offloading failures and energy use, outperforming methods focusing on single metrics or static environments. [24]

Comparing these approaches, Wang et al.'s use of a game-theoretic framework ensures fairness and efficiency but may struggle with scalability in highly dynamic environments where vehicular density fluctuates frequently. [40] Alam et al.'s integration of the Hungarian algorithm within a multi-agent deep reinforcement learning context highlights the importance of cooperative task management. Yet, the dependency on vehicle cooperation can introduce latency issues in less cooperative scenarios. [2] Lin et al.'s focus on joint optimization provides a comprehensive solution, but the complexity of their model might result in higher computational overhead, which could limit real-time applicability. [24]

3.1.2 Optimization Techniques

Optimization techniques are crucial in managing the complex trade-offs in VEC systems, such as balancing energy consumption with latency. Mao et al. present an online joint radio and computational resource management algorithm for multi-user mobile-edge computing MEC systems. The algorithm minimizes mobile devices' long-term average weighted sum power consumption and the MEC server while maintaining task buffer stability. The proposed method optimizes CPU-cycle frequencies, transmit power, and bandwidth allocation using the Gauss-Seidel method, and it features a delay-improved mechanism. The performance analysis indicates a tradeoff between power consumption and execution delay, validated through simulations. [30]

Ibn-Khedher et al. propose an edge computing-assisted autonomous driving framework using Artifical Intelligence (AI). Their architecture features a centralized cloud computing layer for non-real-time tasks, a distributed network edge layer for processing offloaded Virtual Network Function (VNF), and an autonomous vehicles layer. The communication protocol includes edge autopilot slicing, resource discovery in connected edge nodes, and optimized VNF component placement. They employ Integer Linear Programming (ILP) and DRL for resource allocation optimization, demonstrating significant latency and resource utilization improvements through their proposed algorithms OVEAP and DVEAP. Key performance indicators include Total Edge Servers Utilization (TESU), Total Edge Servers Allocation Time (TESAT), and Successfully Allocated Edge Autopilots. [18]

Mao et al.'s method highlights the balance between power consumption and execution delay, which is critical for mobile devices but might not fully address the dynamic

nature of vehicular environments. [30] Ibn-Khedher et al.'s framework, with its focus on autonomous driving, provides a more tailored solution for vehicular networks. Yet, the complexity and reliance on multiple algorithms (ILP and DRL) could introduce significant computational overhead and might not scale well with increasing vehicular density. [18]

3.1.3 Task Assignment and Learning Methods

Rejiba et al. propose a novel approach for computation task assignment in VEC using online learning combined with neighbor advice. In this model, tasks are offloaded from vehicles to nearby vehicles with greater computational capacity, managed by RSUs. The RSUs use online learning to assess the computational capacity of Vehicle-Based Fog Nodes (V-FNs) and provide advice to neighboring RSUs, enhancing decision-making. [35]

Using a Contextual Multi-Armed Bandit (CMAB) learning algorithm, the study optimizes task offloading decisions, including a fixed budget for receiving advice. Evaluations compare independent RSUs, unlimited budget RSUs, limited budget RSUs, and an oracle with prior knowledge. Results show that RSUs using neighbor advice significantly improve learning performance and reduce cumulative task execution delays compared to independent learning. [35]

This approach complements Shah et al.'s distributed control plane architecture for MEC-enabled vehicular networks, focusing on scalability and latency reduction through distributed Software-defined Networking (SDN) controllers. [3] It also aligns with Alkaabi et al.'s review of handover strategies in MEC, addressing mobility and resource management challenges. [4] Rejiba et al. contribute to VEC by integrating learning and advice mechanisms, enhancing task offloading and resource management in dynamic vehicular networks. [35]

Compared to Alam et al., who used a multi-agent DRL-based Hungarian algorithm for task offloading in Internet of Vehicles (IoV) focusing on latency and energy without joint optimization, Lin et al.'s approach comprehensively addresses offloading failure and energy consumption. [2, 24] Similarly, Grislin-Le Strugeon et al.'s MAS for fair computation offloading in VEC emphasizes resource allocation fairness but lacks joint optimization of failure rates and energy. [13]

Rejiba et al.'s use of online learning and neighbor advice introduces a collaborative element to task offloading, which can significantly improve decision-making in dynamic environments. However, relying on neighboring RSUs for advice might introduce latency and require high synchronization and cooperation among RSUs, which can be challenging in real-world deployments. [35]

3.1.4 VEC Offloading Simulation

Simulation tools are essential for evaluating task offloading strategies in VEC environments. These tools help analyze parameters such as energy consumption, latency, and resource

allocation.

IoT and General Edge Computing

iFogSim and CloudSim are frameworks designed for IoT, edge, and cloud computing environments. iFogSim focuses on optimizing energy consumption and latency, [14] while CloudSim models large-scale cloud infrastructures. [7] However, these tools are not tailored for the high mobility contexts typical of vehicular networks.

Vehicular Network Simulations

For high mobility scenarios, SUMO and ns-3 are more appropriate. Simulation of Urban MObility (SUMO) enables microscopic traffic simulation, which is critical for implementing and evaluating Vehicle-to-Vehicle (V2V) and V2I communication approaches. [27] ns-3 supports the simulation of network protocols and communication strategies in dynamic settings, making it ideal for VEC. [36]

VEC Task Offloading

Recent studies have developed specific simulations for VEC environments to address the unique challenges of MEC. Ouarnoughi et al. developed a tool for task offloading in intelligent transportation systems, modeling edge (vehicle), fog (RSU), and cloud layers. Their simulation involves vehicles on a straight road using a greedy algorithm for relay node selection, enhancing task processing efficiency [33]. Liu et al. utilized realistic vehicular mobility traces from the Europarc roundabout in France to simulate vehicular patterns, providing a realistic context for evaluating task offloading and network performance [26]. Ju et al. simulated a complex intersection scene with vehicles forming V2V links, using parameters from 3GPP standards to model the channel, traffic, and vehicle models. This helps in understanding offloading effectiveness at different processing levels [19].

Agent-Based Resource Management

MAS-based simulations leverage intelligent agents to manage and optimize task offloading and resource allocation. Rejiba et al. conducted simulations in a random network topology resembling a campus map, where mobile devices move along predefined paths, helping to understand the impact of task offloading strategies [35]. Grislin-Le Strugeon et al. used the NetLogo tool to model agent behaviors and evaluate resource allocation with and without knowledge sharing between RSUs. Their simulations highlighted the impact of agent communication on offloading efficiency [13]. Wang et al. simulated task offloading across vehicle (edge), RSU (fog), and data center (cloud) levels using NVIDIA Tesla GPUs. They focused on the trade-offs between on-board processing and offloading, depending on the computational load of tasks [40].

3.2 Handover coordination in Vehicular Networks

This section focuses on the strategies for handover coordination in vehicular networks, focusing on distributed architectures, decision-making processes, and the challenges involved in ensuring seamless connectivity and service continuity.

3.2.1 Distributed Architectures and Strategies

Shah et al. propose a distributed control plane architecture for handover management in MEC-enabled vehicular networks, addressing the challenges of ultra-reliable and low-latency communications in dynamic environments. Their approach divides the network into SDN-based domains, each managed by multiple distributed controllers that cooperate to handle vehicle mobility and service requests. This architecture enhances scalability, reduces congestion, and ensures consistent network performance in terms of latency and throughput, as demonstrated by Mininet-WiFi simulations. The handover process, triggered by deteriorating RSS, includes SHO and HHO, each with different impacts on QoS. The MEC Orchestrator (MEO) manages state relocation during handovers, where it selects MEC hosts based on latency and resource availability, ensuring live migration of sessions without interruption. [3]

Alkaabi et al. review handover strategies in MEC, emphasizing the importance of SDN-based architectural approaches. They discuss both centralized and distributed architectures for handover management. In centralized systems, a single SDN controller handles handovers, leading to potential delays in large networks. As described by Shah et al. [3], distributed architectures combine multiple controllers with centralized control logic, allowing faster response times and better handling of extensive service mobility. The handover process, initiated based on RSS and managed by the MEO for state relocation, ensures service continuity [4], aligning with Shah et al.'s approach.[3]

Bao et al. introduce the Follow Me Fog (FMF) framework, designed to support seamless handover timing schemes among different computation access points in a fog computing environment. FMF includes a job pre-migration mechanism, which preemptively migrates computation jobs based on the monitoring of RSS. This approach minimizes service interruptions during handovers, which is crucial for time-sensitive applications. The evaluation results show that FMF can significantly reduce latency, achieving a 36.5% reduction in their experiments. By addressing the gap in efficient handover timing, FMF ensures continuity and low-latency performance in mobile IoT applications, which is particularly relevant for vehicular networks requiring robust handover solutions. [6]

Alkaabi et al. highlight the challenges in MEC handovers, including mobility, migration, synchronization, security, QoS, energy consumption, heterogeneity, scalability, and the lack of standardized protocols. Their review underscores the need for advanced handover management strategies, such as those proposed by Shah et al. and Bao et al., to achieve seamless connectivity and enhanced service quality in MEC-enabled vehicular networks. [4]

Shah et al.'s distributed control plane architecture effectively addresses scalability and congestion through multiple distributed controllers. However, it introduces synchronization challenges and potential points of failure. [3] Alkaabi et al.'s review contrasts centralized and distributed architectures, noting that centralized systems are simpler but suffer latency and scalability issues in large networks. [4] In contrast, distributed systems, as proposed by Shah et al., improve scalability and latency management but increase complexity and synchronization challenges. [3] Bao et al.'s FMF framework offers a practical solution for seamless handovers in fog computing, ensuring low latency and service continuity, which is essential for VEC applications. [6]

3.2.2 Handover Decision Making

Ahmed et al. [1] introduce a cooperative agent-based Vertical Handover (VHO) scheme for heterogeneous networks. This approach integrates intelligent agents in mobile nodes Mobile Nodes (MNs) and APs to predict and manage handovers between different communication technologies, aiming to reduce latency and packet loss.

The key contributions of their work include using agents to collect and process information necessary for handovers. By cooperating, agents in MNs and APs enable efficient decision-making and network selection. The make-before-break (SHO) strategy ensures service continuity during handovers, and the Multi-Criteria Decision-Making (MCDM) process uses parameters like RSS, user preferences, cost, and bandwidth through Grey Relational Analysis (GRA). [1]

Their simulations show that this scheme significantly reduces decision-processing delay compared to centralized methods. It also decreases the handover blocking rate, improving service continuity, and achieves lower packet loss during handovers, indicating higher efficiency. The SHO mechanism further reduces end-to-end delays, enhancing overall network performance. [1]

Ahmed et al.'s cooperative agent-based VHO scheme reduces latency and packet loss using intelligent agents and the SHO strategy, enhancing decision-making and network performance. However, relying on multiple agents introduces overhead and complexity, especially in dynamic environments. Additionally, while reducing end-to-end delays, the SHO strategy requires significant computational resources, potentially limiting scalability. [1]

3.3 MAS Integration in VEC

This section examines the integration of MASs in VEC, focusing on communication patterns, task management, and optimization methods. It highlights how MASs can enhance VEC systems by improving communication efficiency and task coordination.

3.3.1 Communication Patterns in MASs

Gutiérrez et al. present a framework for detecting unbalanced communication patterns in MASs. They categorize agents into overloaders, overloaded, overloaded-overloaders, isolated, and regular to identify communication bottlenecks. The framework includes performance metrics across global, role, and individual scopes. Global metrics compare an agent's messages to the total system messages, role metrics compare an agent's messages within its role, and individual metrics focus on the sent-to-received message ratio. [15]

Empirical studies in crisis management and cinema ticketing validate the framework, showing a correlation between metrics and system performance, especially response times. The framework can guide MAS communication design to improve performance. Integrating this framework into VEC can enhance communication efficiency during handovers, ensuring balanced communication and more reliable operations. [15]

Grislin-Le Strugeon et al. [13] introduce a MAS designed to make computation offloading fairer in VEC. In their setup, intelligent agents at both the vehicle (edge) and RSU (fog) levels manage task distribution, which is essential for the real-time needs of AD. They propose offloading tasks to fog and cloud resources to tackle AD's heavy data processing demands. Vehicle agents oversee tasks and assess system load, while RSU agents handle offloading requests and ensure fair resource allocation. They prioritize requests from less-served vehicle agents and share knowledge among RSU agents to maintain service continuity. Their simulations show significant improvements in efficiency and fairness, underscoring the importance of agent communication in dynamic VEC networks. [13]

Ouarnoughi et al. [33] build upon the previous work by developing a comprehensive simulation tool for task offloading in Intelligent Transportation Systems (ITS) to model edge (vehicle), fog (RSU), and cloud layers. The previously described fairness metric measures the proportion of resources allocated to the least and most served agents over time, ensuring balanced access to remote computing resources. However, their approach has some notable shortcomings. The study does not consider the costs of switching between RSUs, which can impact overall system efficiency. Additionally, the focus on fairness does not consider the potential overhead or latency introduced during handovers, which is critical for real-time applications. Further, the assumption that a vehicle is always connected to the nearest RSU may not hold true in dynamic and dense urban environments. [33]

In comparison, Gutiérrez et al.'s framework for detecting unbalanced communication patterns provides a valuable tool for improving system performance by identifying and addressing communication bottlenecks. However, the implementation complexity might limit its practical application in highly dynamic vehicular environments. [15] Grislin-Le Strugeon et al.'s MAS emphasizes fairness in task distribution, which enhances efficiency but does not address potential overhead and latency issues during handovers. [13] Ouarnoughi et al.'s simulation tool extends the work of Grislin-Le Strugeon et al. by modeling edge, fog, and cloud layers. However, it overlooks the costs of switching between RSUs and the assumption of constant connectivity to the nearest RSU. [15]

3.3.2 Task Management and Optimization

Carlier et al. propose a new approach for managing the processing capabilities of vehicle systems by leveraging MASs. Their framework integrates IoT-A architectures, placing agents close to the hardware, and Avatar architectures, which virtualize hardware in cloud and edge computing environments. This dual approach, called EdgeAgent, aims to enhance vehicle management by effectively distributing computational tasks across available resources while considering future vehicle requirements. [9]

Huang et al. introduce an energy-efficient offloading decision-making scheme for VEC to meet the growing demands of 5G vehicular networks by offloading computation-intensive and delay-sensitive applications from vehicles to the network edge. [17] Their primary contributions include a dynamic task offloading decision model that minimizes a utility function accounting for energy consumption and packet drop rates. Tasks are categorized into local and flexible subtasks, optimizing the offloading process. Additionally, a computation resource allocation scheme distributes VEC resources based on each vehicle's computation intensity and queue status. They propose a Lyapunov-based dynamic offloading decision algorithm that integrates task offloading with resource allocation, ensuring queue stability. Their simulation results show significant reductions in energy consumption and packet drop rates, enhancing overall network performance compared to existing methods. [17]

Xu et al. [41] propose V2X-COM, a method using V2X communication for computation offloading in edge computing. V2X-COM addresses latency and resource utilization by employing NSGA-III to generate balanced offloading strategies and using Simple Additive Weighting (SAW) and MCDM to select the optimal strategy. Experimental results show V2X-COM effectively reduces latency and improves resource utilization compared to traditional methods. [41]

Comparing these approaches, Carlier et al.'s EdgeAgent architecture provides a comprehensive framework for managing processing units in a multi-agent context but may introduce significant complexity in real-world deployment. [9] Huang et al.'s energy-efficient offloading decision-making scheme reduces energy consumption and packet drop rates, offering a balanced solution for 5G vehicular networks. However, its reliance on a Lyapunov-based algorithm may limit its adaptability to highly dynamic environments. [17] Xu et al.'s V2X-COM method emphasizes joint latency and resource utilization optimization, presenting a robust alternative to Huang et al.'s energy-focused approach. While both methods use dynamic task offloading models, Xu et al.'s strategy might be more effective when latency and resource utilization are critical. [41]

3.4 Potential and Integration Challenges of MASs in VEC

This section explores potential use cases and challenges in integrating MASs into VEC, considering vehicular networks' unique requirements and constraints.

3.4.1 Use Cases and Potential Benefits

1. **Enhanced Task Coordination and Optimized Resource Allocation**: The integration of MASs into VEC can significantly enhance task coordination and resource allocation. Agents can autonomously decide task distribution among vehicles and edge nodes, ensuring optimal use of available resources in real-time. [13, 17] This capability is crucial in dynamic environments where conditions can change rapidly.

2. **Improved Communication Efficiency and Adaptive Traffic Management**: MASs can improve communication efficiency by managing communication patterns and reducing bottlenecks. Intelligent agents can predict traffic flow and optimize routing to minimize congestion. [15, 33] This leads to more efficient data transmission and smoother traffic flow, reducing overall latency and improving QoS.

3. **Enhanced Fault Tolerance and Resilience**: MASs can enhance the resilience of VEC systems by enabling distributed fault detection and recovery. Agents can quickly identify and respond to network failures or performance issues, ensuring the system remains operational despite adverse conditions. [2] This distributed approach to fault tolerance helps maintain system stability and reliability.

3.4.2 Integration Challenges

1. **System Complexity and Scalability**: Integrating MASs into VEC introduces significant complexity, due to the need for sophisticated coordination mechanisms among numerous intelligent agents. This complexity can make it challenging to ensure that all agents work harmoniously without conflicts. Additionally, as the number of vehicles and edge nodes increases, ensuring that the MAS can scale effectively without degrading performance becomes a critical concern. [3, 9]

2. **Agent Communication Overhead and Robust Algorithms**: The continuous exchange of information among agents can lead to substantial communication overhead. Managing this overhead is essential to maintain the benefits of MASs, especially in bandwidth-constrained vehicular networks. [15] Furthermore, the dynamic nature of vehicular environments requires robust and adaptive algorithms capable of handling frequent changes in network topology, varying traffic conditions, and fluctuating resource availability. [41]

3. **Security, Privacy, and Standardization**: The integration of MASs into VEC must address security and privacy concerns. Agents must operate securely, protecting sensitive data from unauthorized access and ensuring secure communications. [3] Moreover, the lack of standardized protocols for MASs in VEC can hinder integration efforts. Ensuring interoperability between different MASs implementations and existing vehicular network protocols is crucial for seamless integration. [4]

4. **Heterogeneity of Devices**: VEC environments consist of heterogeneous devices with varying capabilities. Designing MASs that can effectively operate across diverse devices, each with different computational and communication capabilities, is challenging. [17] Ensuring compatibility and efficient operation across these heterogeneous devices is essential for successful integration.

Despite these challenges, integrating MASs into VEC promises to create more efficient, adaptive, and resilient vehicular networks. Continued advancements in this field will be crucial for realizing MASs' full potential in enhancing VEC systems' performance and reliability.

CHAPTER 4

Approach: Agent-based Handover Coordination in VEC

This chapter details the development of the Agent-based RSU Handover Coordination (ARHC) strategy for handover management in VEC environments. It begins by discussing the specific motivations for adopting a decentralized, RSU-based MAS approach. Following this, the chapter outlines the key objectives guiding the research, with a specific focus on the ARHC strategy as the primary contribution of this thesis. Subsequently, it introduces the network model and agent architecture that form the foundation of the ARHC strategy and provides an overview of the custom simulation environment used to assess its effectiveness. The evaluation methodology is also outlined. Details on the actual functioning of the ARHC strategy are provided in Chapter 5, where its technical implementation is thoroughly discussed.

4.1 Motivation and Objectives

4.1.1 Motivation

Efficient handover coordination in VEC is crucial for maintaining high QoS in VEC environments, particularly in scenarios that demand low-latency processing and heavy computational demands, such as AD. Traditional centralized and vehicle-based handover strategies often struggle to balance scalability, resource utilization, and network performance under dynamic conditions. This thesis introduces the ARHC strategy, a decentralized, RSU-based approach that leverages MASs, as detailed in Section 2.4, to optimize handover processes. The goal is to address the limitations of existing methods by enhancing load distribution, reducing unnecessary handovers, and ensuring consistent QoS across dynamic VEC environments.

4.1.2 Objectives

1. **Develop ARHC Strategy**: Create a decentralized handover coordination strategy utilizing RSU-based agents to enhance handover efficiency in VEC environments.
2. **Optimize Resource Utilization and QoS**: Ensure ARHC effectively balances computational loads and maintains high QoS, particularly in scenarios with varying RSU capacities and traffic densities.
3. **Design a Custom Simulation Environment**: Build a simulation framework to evaluate ARHC, incorporating real-world traffic data and variable RSU configurations.
4. **Benchmark ARHC Against Existing Strategies**: Evaluate ARHC's performance relative to traditional centralized and vehicle-based methods, focusing on metrics like handover frequency, QoS, and load distribution.

This approach provides a focused and concise foundation for understanding the ARHC strategy and its relevance within VEC research, with the technical implementation details provided in the subsequent Chapter 5.

4.2 Approaches to Handover Coordination

Effective handover coordination is critical for seamless connectivity and optimal performance in VEC. Various approaches to handover coordination exist, each with its strengths and limitations. This section discusses three primary strategies: centralized, vehicle-based, and RSU-based handover coordination.

The following subsections provide an in-depth examination of these approaches, highlighting the rationale behind selecting an RSU-based strategy for this thesis.

4.2.1 Centralized Handover Coordination

Centralized handover coordination involves a central controller managing the entire network's handover decisions. This approach typically leverages a global network view to optimize handovers for improved overall performance and resource utilization.

Centralized approaches often use a centralized or distributed control plane to manage handovers. While these approaches can achieve efficient network-wide optimization, they do not employ decentralized agent-based strategies, which are the focus of this thesis. Furthermore, comparing centralized approaches to decentralized agent-based approaches can be challenging due to differences in fundamental design and operational principles. [2, 4, 24, 40] Therefore, exploring centralized handover coordination is not within the scope of this thesis, but relevant mentions and discussions can be found in Section 3.2.

4.2.2 Vehicle-based Handover Coordination

Vehicle-based handover coordination is a common approach in VEC systems, where the vehicles handle the decision-making. As explored by Bao et al. [6], Ouarnoughi et al. [33], and Grislin-Le Strugeon et al. [13], this method relies on metrics like RSS from multiple RSUs to guide decisions on when a vehicle should switch connections.

Each vehicle monitors its signal strength in this setup and decides when to initiate a handover to improve its own QoS. However, because each vehicle makes these decisions independently, they don't always result in the best performance for the entire network. For instance, the role of an RSU is often limited to accepting or rejecting a vehicle's handover request, as discussed by Bao et al. [6], which limits the overall coordination across the network. For a more detailed explanation of signal strength-based handovers, refer to Section 2.2.

A typical strategy used in vehicle-based handovers is based on RSS, with added mechanisms like hysteresis to prevent frequent, unnecessary handovers [6]. Alternatively, adaptive thresholds that consider the expected stability of the connection can be used [34]. While these methods help maintain a stable connection for individual vehicles, they focus mainly on local conditions rather than the overall network.

Similar RSS-based strategies are common in cellular networks, where signal strength determines handover timing [34]. While this approach can work well for individual vehicles, it can cause problems when many vehicles try to connect to the same RSU simultaneously, leading to congestion and reduced network performance.

In conclusion, vehicle-based handover coordination focuses on local decision-making to improve individual QoS. However, this approach can overlook the broader needs of the network, suggesting that more integrated, network-aware strategies could lead to better overall performance.

4.2.3 ARHC Strategy

Our ARHC strategy introduces a shift in handover decision-making from vehicles to the RSUs within the fog computing layer. These RSUs are interconnected via high-speed, low-latency wired networks, enabling real-time sharing of vital data such as vehicle position, processing load, and latency metrics. This interconnectedness allows RSUs to make informed handover decisions, optimizing QoS across the network while minimizing wireless network congestion.

In ARHC, RSUs monitor continuous updates from vehicles, including their position and speed. When a vehicle approaches the edge of an RSU's coverage, the RSU preemptively coordinates a handover to the optimal neighboring RSU, ensuring uninterrupted service. This proactive approach is crucial for maintaining seamless connectivity, particularly for latency-sensitive tasks such as AD. The technical details of this handover process, including how RSUs monitor vehicles, calculate load, and coordinate handovers, are

elaborated in Chapter 5, where the decision-making algorithms and communication protocols are thoroughly described.

Additionally, RSUs can simultaneously manage multiple vehicles, making decisions that benefit the entire network. For example, if an RSU anticipates an overload due to high traffic, it can preemptively transfer some vehicles to neighboring RSUs, ensuring balanced load distribution and optimal resource utilization. This decentralized strategy enhances system resilience and responsiveness, allowing the network to quickly adapt to dynamic conditions, thereby avoiding delays and bottlenecks associated with centralized control.

Advantages of the ARHC Strategy

The ARHC strategy leverages a RSU-based, agent-oriented approach that provides several notable advantages in VEC:

- **Reliability**: The decentralized nature of our approach enhances reliability by eliminating single points of failure, ensuring continuous operation even when individual RSUs face issues.
- **Resilience**: The system is highly adaptive, dynamically responding to changing conditions such as fluctuating traffic loads or RSU outages, thereby increasing overall resilience.
- **Scalability**: The approach is designed to scale efficiently with increasing numbers of vehicles and RSUs, making it suitable for large, complex vehicular networks.
- **Optimized Resource Utilization**: By distributing computational tasks based on current load and capacity, the system efficiently uses available resources, preventing any single RSU from becoming a bottleneck.
- **Low Latency**: Localized decision-making at the RSU level reduces communication delays, which is crucial for time-sensitive applications such as AD.
- **Reduced Communication Overhead**: The strategy minimizes unnecessary handovers and related signaling, leading to more efficient use of network resources and reduced communication overhead.
- **Improved QoS**: Through effective handover coordination and resource management, the system consistently maintains high QoS, ensuring a reliable and smooth user experience across all connected vehicles.

4.3 Network model

To understand the approach presented in this thesis, it is essential to comprehend the network model, including the participants, their placement, and core functionalities.

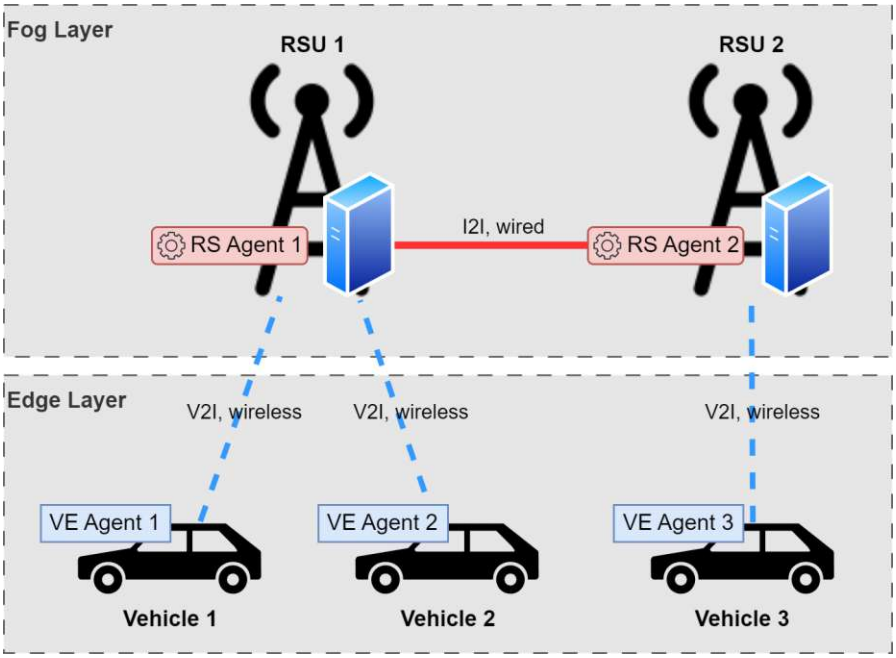


Figure 4.1: Network layers illustrating the placement of RSUs and vehicles. RSUs are equipped with MEC servers and communicate with vehicles to facilitate handover coordination

The network model relevant to this thesis consists of two main layers: the edge and the fog layer. The edge layer contains the vehicles, whereas RSUs are located inside the fog layer. Although the cloud layer is part of the broader network model, it is irrelevant to this thesis due to the strict latency requirements of the tasks we focus on, as outlined in Section 1.1. These layers are depicted in Figure 4.1 and further described in Section 2.1.2.

Our network model integrates RSU-based and in-vehicle agents to streamline handover processes and optimize resource allocation. This agent-based architecture, detailed in Section 4.4, equips vehicles and RSUs with dedicated agents, as shown in Figure 4.1. This approach contrasts with traditional models by embedding intelligence directly within network components. The specific interactions and protocols of these agents are elaborated in Chapter 5.

Like Lin et al. [24], the RSUs are deployed at fixed locations along the roads, each outfitted with a MEC server with specific computing capabilities and integrated with a wired connection. This enables RSUs to perform computations on behalf of the vehicles, as outlined in Section 2.2, within their coverage radius.

Vehicles are equipped with onboard computation devices that handle a portion of their computational tasks locally, adhering to low-latency constraints. In this thesis, however, we simplify the model by assuming that a static portion of the vehicle's computational load is processed onboard, with the remaining load offloaded to the RSUs. This approach

ensures that the focus remains on tasks handled by the RSUs, excluding cloud offloading due to latency requirements. The randomized load, adjusted after onboard processing, reflects more realistic vehicular environments while keeping the model manageable for this study. Although offloading decision-making is critical and thoroughly discussed in works by Grislin-Le et al. [13], Huang et al. [17], and Lin et al. [24], it is not the primary focus here.

The RSUs are interconnected via a wired network, facilitating efficient communication and minimizing network congestion. This inter-RSU communication, known as Infrastructure-to-Infrastructure (I2I) communication, is depicted by the solid red line in Figure 4.1. I2I communication is critical for optimizing task offloading and managing handovers, allowing RSUs to share load information and coordinate actions seamlessly.

Vehicles communicate wirelessly with nearby RSUs, focusing solely on V2I communication, represented by the dashed blue lines in Figure 4.1. This approach reduces complexity, as V2I communication scales with the number of vehicles and RSUs, unlike V2V communication, which scales quadratically with the number of vehicles. This setup aligns with models presented by Bao et al. [6], Lin et al. [24], and Ouarnoughi et al. [33]

No realistic communication model like bandwidth or RSS is used for simplicity. Instead, a fixed range and known positions of RSUs and vehicles are used for distance calculations. No network, communication, or infrastructure-related crashes are considered in the simulation. Security concerns are not addressed in this study. This thesis does not directly address cellular networks and protocols but builds upon technologies like 5G and WIFI 802.11p. Instead, it abstracts these underlying technologies to focus on effectively coordinating handovers between RSUs.

Understanding the network model is crucial for comprehending the proposed approach, as it supports the communication and coordination strategies essential for efficient handover management in vehicular environments.

4.4 Agent architecture

As detailed in Section 4.3, this thesis focuses on the edge and fog layers within the network model, where our agent-based approach is deployed. The architecture includes two types of agents: In-Vehicle (VE) agents embedded in each vehicle and RoadSide (RS) agents located at each RSU, both integrated into the network model, as shown in Figure 4.1. VE agents run on the onboard computing systems within vehicles, utilizing the vehicle's hardware to monitor status, make offloading decisions, and communicate with RS agents. Meanwhile, RS agents operate on the MEC servers within the RSUs, strategically placed along the roadside to manage offloaded computational tasks, coordinate handovers, and optimize resource utilization across the network.

These agents are organized in a flat hierarchy within their respective categories, meaning all VE agents operate at the same level without any internal ranking, and the same applies to the RS agents. This interaction is crucial for coordinating actions and achieving

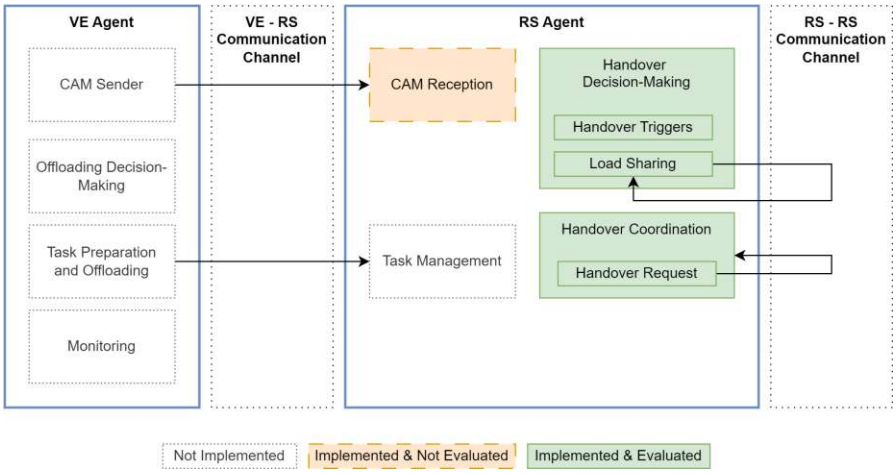


Figure 4.2: Logical components of VE and RS agents. Components with solid lines are implemented and evaluated, components with dashed lines are implemented but not evaluated, and components with dotted lines are not implemented

the system's overall objectives, aligning with the frameworks established by Ouarnoughi et al. [33] and Grislin-Le Strugeon et al. [13].

Figure 4.2 illustrates the components of each agent. The VE agent includes several modules: Monitoring, offloading decision-making, task preparation and offloading, and a Cooperative Awareness Message (CAM) sender. On the other hand, the RS agent contains handover coordination, task-offloading management, and handover decision-making modules, including load sharing and handover triggers as sub-components. The figure also highlights the communication mechanisms between VE and RS agents, as well as inter-RS agent communication. The implementation details of these modules, particularly the decision-making processes and communication protocols that govern the interaction between VE and RS agents, are discussed in Chapter 5.

The diagram uses specific visual conventions to indicate the implementation status of these components:

- Dotted lines and a white (unshaded) background denote components that are not implemented.
- Dashed lines with an orange shading indicate implemented but not evaluated components.
- Solid lines with green shading represent components that have been fully implemented and evaluated.

To clarify the terminology used in this thesis: "Vehicle" and "RSU" refer to physical devices, such as cars and roadside units. In contrast, VE agent and RS agent refer to

the software agents embedded within these devices. These agents are responsible for communication, decision-making, and coordination tasks within the network. We discuss the physical devices whenever we mention a vehicle or RSU. In contrast, references to VE or RS agents specifically address the logical entities that manage the intelligent operations of the system.

4.4.1 In-vehicle (VE) Agent

As mentioned, each vehicle hosts a VE agent. According to Ouarnoughi et al. [33], the primary functions of a VE agent include supervision and monitoring of the vehicle's system activities and surroundings, task evaluation, offloading decision-making, task offloading, and communication with RS agents.

In the context of this thesis and the proposed ARHC strategy, the critical aspects involve monitoring activities and communicating with RS agents. Although offloading decision-making is essential for effectively utilizing MAS, it introduces considerable complexity. This complexity is unnecessary for evaluating ARHC, as outlined in Section 4.3.

On a meta-level, the VE agents also manage task offloading in collaboration with the corresponding RS agents. This process is assumed to occur in the background because the RS agents can access current and historical data regarding offloaded tasks and their complexities. By monitoring the amount and complexity of offloaded tasks, RS agents can effectively manage this aspect without further detailed investigation into the offloading decision-making processes. This simplification allows us to treat handover coordination as independent from the offloading process, thereby reducing unnecessary complexity and focusing on the primary objectives of the thesis.

In this thesis, one of the most crucial responsibilities of VE agents is to inform RS agents of their current location periodically. This is achieved using a mechanism developed by the European Telecommunications Standards Institute (ETSI) for ITS, known as the Cooperative Awareness Message (CAM). Through this mechanism, vehicles periodically transmit beacon messages containing information about their position, direction, speed, and other relevant data [10, 39]. According to the ETSI EN 302 637-2 standard [10], each vehicle sends this message 1 to 10 times per second.

4.4.2 Roadside (RS) Agent

The RS agents play a crucial role within the MAS architecture by managing both handover coordination and task offloading. Their strategic position within the network allows them to facilitate seamless transitions and maintain optimal system performance.

Receival of CAMs

As outlined in the previous section, RS agents receive CAMs from currently connected vehicles. These messages provide real-time updates on vehicle positions, directions, speeds, and other relevant data, enabling RS agents to monitor and manage vehicle

connections effectively. The periodic reception of these updates is essential for the RS agents to understand the vehicular network's dynamics accurately. In turn, they allow RS agents to make informed handovers and task-offloading decisions, ensuring smooth network operations.

Task Offloading Management

Task offloading is a collaborative process managed by both VE and RS agents. While detailed offloading decision-making processes are outside the scope of this thesis, it is essential to note that RS agents have inherent access to current and historical data on offloaded tasks and their complexities. See the previous Section 4.4.1 on VE agents for reasoning.

By monitoring the volume and complexity of these tasks, RS agents can manage task loads effectively without requiring further detailed investigation into the offloading decision-making processes. This approach simplifies the overall system by treating offloading processes as background operations.

General Strategies for Handover Coordination

Handover coordination involves a MCDM process centered on three relevant dimensions: connectivity, capacity, and communication.

1. **Connectivity**: This dimension refers to the RS agent's ability to maintain a stable and high-quality connection with a vehicle. Typically influenced by signal strength and distance, connectivity ensures that vehicles remain within the optimal range of an RSU for seamless data transmission. This thesis simplifies connectivity by assuming that an RSU can provide 100% QoS within a specified range. While in real-world scenarios, connectivity might also depend on factors like bandwidth, network congestion, and variable latencies due to different tasks, these are abstracted in this study to maintain simplicity and focus on the model's core elements. This assumption is made to streamline the analysis and emphasize other aspects of the network. Handovers due to connectivity issues occur when a vehicle exits the RSU's range, where "range" is defined as the area within which full QoS is guaranteed.

2. **Capacity Management**: In this thesis, capacity management focuses on how RSUs handle offloaded computational tasks from vehicles. Vehicles process part of their load locally, offloading the excess to RSUs, which manage the aggregated demands from multiple vehicles. The computational load varies dynamically based on environmental factors and task complexity, requiring RSUs to adapt to these changes efficiently. This approach emphasizes the importance of dynamic capacity allocation to maintain high QoS and prevent RSU overloads in vehicular networks, as discussed by Ouarnoughi et al. [33].

3. **Communication Management**: Communication involves the data exchange between vehicles and RSUs, as well as among RSUs. This includes managing the rate and volume of communication, such as the frequency of CAMs sent by vehicles and sharing load information among RS agents through I2I communication. Effective communication management is crucial for balancing network efficiency and reducing congestion, ensuring that necessary information is shared without overwhelming the network. This supports optimal decision-making and coordination among the agents involved in the handover process.

These strategies are integral to optimizing network performance and resource utilization, ensuring that connectivity, capacity, and communication are effectively managed to support seamless handover processes and maintain high QoS across the vehicular network.

Handover Coordination and Execution

RS agents monitor vehicles within their range through periodic updates received via CAMs. When a vehicle is detected to be leaving the range, the RS agents initiate handovers to the most suitable neighboring RSUs, factoring in connectivity and capacity. Additionally, RS agents perform load-balancing by preemptively transferring vehicles to neighboring RSUs with lower loads, optimizing the overall QoS in the local system.

Effective handover coordination relies heavily on inter-RS-agent (I2I) communication. RS agents proactively exchange load status and connectivity information to trigger handovers, ensuring a balanced distribution of tasks across the network. Prioritization is crucial, mainly when a vehicle exits the RSU's range or when an RSU nears its capacity limit. This approach is consistent with strategies observed in other studies, such as those by Grislin-Le Strugeon et al. [13], who investigated the impact of communication and load-sharing history on offloading fairness.

The handover execution involves migrating the vehicle's computation state from the source RSU to the target RSU. This state includes data necessary for continuing the offloaded tasks seamlessly. To maintain simplicity, this thesis assumes a constant state size for all vehicles, facilitating the focus on the frequency and effectiveness of handovers rather than the actual data transfer. Seamless state migration is crucial for maintaining continuity in offloaded tasks and ensuring that QoS standards are met throughout the handover process, as outlined in Section 2.2.3.

RS agents are central to the agent-based architecture, managing both handover coordination and task offloading. They ensure efficient handover processes, optimal load balancing, and overall system performance by leveraging real-time CAM data and robust inter-RSU communication. This strategic approach simplifies the system's evaluation, allowing a concentrated analysis of the primary objectives of this thesis.

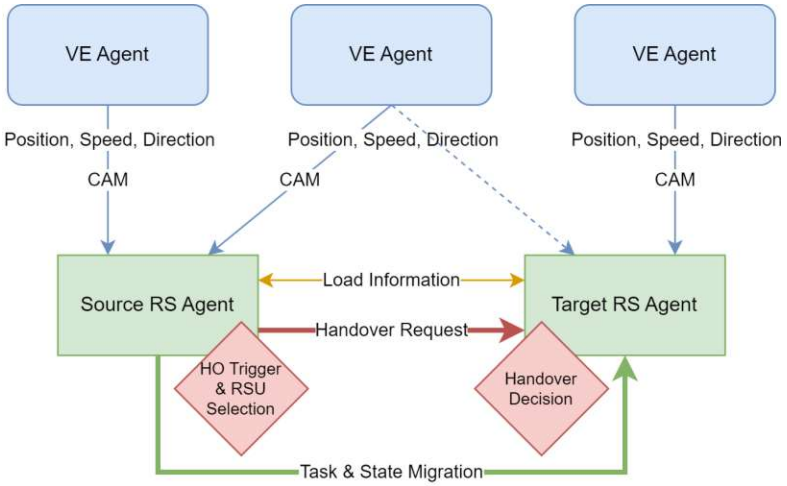


Figure 4.3: Overview of Communication Flows between VE and RS Agents in the ARHC Strategy

4.4.3 Overview of Agent Communication

Following the introduction of VE and RS agents, it is essential to understand how these agents communicate to manage handovers and optimize resource allocation within the ARHC strategy. Figure 4.3 provides a visual overview of these interactions, highlighting the flow of information essential for maintaining seamless operation.

In the ARHC strategy, VE agents transmit CAMs to their connected RS agents. These messages contain crucial data on the vehicle's position, speed, and direction, enabling the RS agents to monitor vehicle status in real-time. The consistent flow of CAMs is depicted by the blue arrows in Figure 4.3, emphasizing the continuous communication between VE and RS agents.

In addition to receiving CAMs, RS agents engage in I2I communication with neighboring RSUs, shown by the orange arrows in the figure. This exchange of load information allows RS agents to make informed decisions during handovers, such as when a vehicle moves towards the edge of a coverage area or when an RSU approaches its load capacity.

When a handover is necessary, the source RS agent evaluates potential target RSUs based on factors such as vehicle trajectory and RSU load. If the conditions are favorable, a handover request is sent to the target RS agent (red arrow). Upon acceptance, the vehicle's connection and computational tasks are transferred, as indicated by the green arrow, ensuring that the handover is executed smoothly and without disrupting service.

Figure 4.3 thus encapsulates the essential communication flows within the ARHC strategy, illustrating how coordinated actions between VE and RS agents maintain high QoS and efficient resource use across the network.

4.5 Novelty of the Approach

The novelty of this thesis lies in integrating MASs within VEC, particularly focusing on the ARHC strategy. This area remains largely unexplored in current research, especially when addressing handover coordination in dynamic vehicular environments. This thesis offers significant contributions through the development of the ARHC strategy and a custom simulation environment tailored for comprehensive evaluation.

Key novel aspects of this thesis include:

- **ARHC Strategy for RSU-based Handover Coordination**
  The ARHC strategy introduces a decentralized, agent-based approach to handover coordination at the fog layer, moving beyond traditional centralized coordination or vehicle-based methods. This strategy leverages RSUs with embedded agents to manage handovers more effectively, optimizing decision-making based on real-time data exchange and vehicle trajectories.
- **Reduction of Unnecessary Handovers through Predictive Analytics**
  The strategy incorporates predictive analytics to minimize unnecessary handovers, reduce communication overhead, and maintain seamless service continuity. This approach ensures that handovers are efficient and occur only when necessary.
- **Adaptive Load Distribution for Enhanced QoS**
  RSUs dynamically balance computational loads using adaptive algorithms, optimizing resource utilization and preventing overloads, as detailed in Section 5.2. This ensures consistent QoS across varying traffic conditions and vehicular densities.
- **Custom Simulation Environment with Real-World Data**
  A tailored simulation environment was developed to evaluate the ARHC strategy rigorously. This environment uses real-world traffic datasets, providing realistic validation of the strategy's effectiveness in managing handovers and optimizing resource allocation.

Compared to existing approaches, the most similar works are those by Ahmed et al. [1] and Ouarnoughi et al. [33]. Ahmed et al. explored using MASs for coordinating handovers in heterogeneous cellular networks, involving agents at both mobile nodes and access points, and targeting SHOs using RSS and specific thresholds for network switching. The novelty of our approach lies in its focus on VEC systems, addressing the particular challenges of offloaded computation and required state migration. Additionally, our decision process relies heavily on RSU coordination for local and global optimization, incorporating RSU computation capacity, which introduces a new dimension to the problem. We also develop a custom simulation environment tailored to VEC systems and use a real-world dataset for validation.

Ouarnoughi et al. investigated setting up a simulation environment for agent-based cooperative computation offloading in VEC, using VE and RS agents, aiming for fairness

in computation offloading without focusing on handover costs. They used the nearest-RSU policy, which will also be used as a benchmark in our evaluation. Unlike their approach, this thesis explicitly addresses the handover process, necessitating the creation of a new simulation environment to handle the intricate communication details of handovers. Additionally, our use of a real-world dataset ensures that the presented data is more applicable to real-world scenarios.

By addressing these novel aspects, this thesis aims to advance the state of VEC handover coordination, ultimately enhancing the performance and reliability of vehicular networks.

4.6 Simulation Environment

4.6.1 Overview of Simulation Goals

The primary aim of this simulation is to test and evaluate handover strategies within VEC using MAS, specifically focusing on the ARHC strategy. The detailed evaluation is presented in Chapter 6. The specific goals of the simulation are as follows:

1. **Assess Handover Strategies**: Evaluate the effectiveness of various task handover strategies to ensure seamless connectivity and low-latency processing in VEC scenarios while upholding high QoS and distributing the load among RSUs evenly.
2. **Mirror Real-World Scenarios**: Develop a simulation environment that closely mirrors real-world VEC scenarios. This involves creating dynamic and controllable environments that accurately reflect real-world vehicular networks' complexities.

To achieve this, we designed a custom simulation environment to address the intricacies of agent-based handover strategies within VEC scenarios, as detailed in Section 5.3. This environment was explicitly developed to address the intricacies of agent-based handover strategies, providing a flexible platform for evaluating the effectiveness of MASs in VEC scenarios. Unlike existing frameworks that may include VEC-specific features or constraints, our approach prioritizes flexibility and generality, enabling broad applicability across various scenarios.

As a result, we designed a new simulation environment that provides a robust foundation for implementing and analyzing task handover strategies, including ARHC, without being constrained by the limitations of pre-existing VEC-specific frameworks, and which that closely resemble real-world VEC conditions.

4.6.2 Simulation Setup

This section overviews the simulation environment, including software tools and frameworks.

Initially, the plan was to use an existing simulation to leverage pre-existing code, such as the one presented by Ouarnoughi et al. [33]. However, during the research phase, it

became evident that existing simulations, especially those employing MAS approaches, are primarily designed for simulating computation offloading. These simulations often incorporate advanced concepts like real-time schedulers to manage progression over time or actual computation on GPUs, which are not necessary for the focus of this thesis - coordinating handovers.

Existing simulations were found to lack the necessary support for evaluating agent interactions, a critical component of this study. Therefore, we opted to develop a new agent-based simulation, utilizing existing tools rather than starting entirely from scratch.

The core of this simulation is built using the Python Mesa framework, presented in [20]. Mesa is a versatile agent-based modeling and simulation platform offering an easily extensible browser-based visualization interface. It facilitates extracting and analyzing information from the model and agents over time, which is crucial for the objectives of this thesis. Mesa aims to provide functionalities similar to well-established MAS simulation frameworks like NetLogo, Repast, or MASON, as described in Section 5.3.1. [20]

The simulation uses distinct time steps with a fixed timespan between them. This interval can be adjusted to match the frequency or fraction of the dataset intervals. This approach divides the vehicle travel time into discrete time slots, making the system quasi-static within each slot while allowing changes across different slots, as detailed in Section 5.3.1, and as described by Lin et al. in [24].

4.6.3 Scenario Development

The scenario development for this thesis centers on routine operations in urban environments where the density and coverage of RSUs with integrated computing capacity are sufficient to meet the QoS requirements for AD tasks or are slightly below the required capacity (in some locations) to test overload behavior. These scenarios balance complexity with manageability by focusing on small areas with intricate street layouts and high-traffic conditions involving numerous vehicles moving simultaneously.

These scenarios are engineered to capture high-fidelity data, recording vehicle positions at least every few seconds. This frequent and precise data collection allows for additional metrics such as speed and orientation, which can also be derived from the detailed location information.

The scenario designer manually configures the placement, range, and capacity of RSUs within the simulation, as outlined in Section 5.3.2. Similarly, vehicle offload load parameters are manually set and vary at the given interval. This manual setup ensures that the simulation environment closely mirrors real-world conditions and provides a robust foundation for testing handover strategies.

The simulation simplifies the environment by focusing on evaluating handover strategies. It assumes fixed RSUs locations with stable capacities and reliable connections, ideal communication conditions, and static offloading of computational tasks. While primarily focused on normal operations, it also explores RSU failures to assess system robustness.

4.6.4 Data collection

Data collection during simulation runs is crucial for evaluating the performance of handover strategies in our VEC environment. The simulation operates on discrete timesteps, providing a structured framework for extracting and analyzing agent and system metrics. This approach ensures detailed temporal resolution, essential for capturing the dynamics of handover processes and their impact on QoS, further described in Section 5.3.3.

Per-Agent Metrics

- **Number of Connected Vehicles**: This metric tracks the number of vehicles connected to each RSU at every timestep. It helps understand the load distribution across the network.
- **Offloaded Load per RSU**: This measures the computational load offloaded to each RSU. By tracking this load, we can assess the capacity utilization and identify potential bottlenecks.
- **Average and Minimal QoS**: These metrics provide insights into the overall performance and the worst-case scenarios for vehicles connected to each RSU. Monitoring QoS at each timestep allows for real-time assessment of network performance and the effectiveness of handover strategies.

System-Wide Metrics

- **Aggregate Metrics**: These include the total number of connected vehicles, overall offloaded load, and average QoS across all RSUs. Aggregating these metrics provides a macro view of network performance and helps identify trends and patterns.

Differentiation between Agent Types

The simulation distinguishes between VE and RS agents. This differentiation allows for more granular analysis of:

- **VE Agent Metrics**: These metrics focus on the vehicle's experience, including connectivity stability, QoS, and the effectiveness of handover decisions.
- **RS Agent Metrics**: These metrics assess the performance and load management of RSUs, including their ability to handle offloaded tasks and coordinate with neighboring RSUs.

This detailed data collection framework supports a comprehensive evaluation of handover strategies by providing raw metrics that can be directly extracted from the simulation. These metrics form the foundation for analyzing the performance and efficiency of different handover coordination approaches in the subsequent evaluation section.

4.7 Evaluation Strategy

4.7.1 Performance Metrics and Benchmarks

This section defines the key performance metrics and benchmarks that will be used to evaluate the effectiveness of the handover strategies in our VEC environment. The evaluation aims to understand the impact of different strategies on coordination efficiency, QoS, and load balancing.

Performance Metrics

1. **Number of Successful Handover Attempts**

   The number of successful handover attempts is a critical metric that counts instances when an RS agent successfully hands over a vehicle to another RS agent. A handover is successful when the target RS agent accepts the request. This metric is crucial for understanding the effectiveness of the handover strategy and the communication overhead involved in state migration.

2. **Number of Failed Handover Attempts**

   This metric tracks instances where a handover attempt is initiated but declined by the target RS agent. Many failed handovers indicate inefficiencies in the handover decision-making process and contribute to unnecessary communication overhead. This metric helps identify areas where the handover strategy can be optimized.

3. **QoS experienced by Vehicles**

   The QoS experienced by vehicles is measured through two sub-metrics:

   - **Average QoS per Timestep**: Captures the average QoS experienced by all vehicles at each timestep, providing a broad view of overall network performance.
   - **Minimum QoS per Timestep**: Records the lowest QoS experienced by any vehicle at each timestep, highlighting worst-case scenarios and potential service degradation points.

   To comprehensively assess the effectiveness of these handover strategies, QoS is analyzed through two key components: distance-based QoS and load-based QoS. These metrics allow us to evaluate how well the network manages spatial and computational resources, ensuring that vehicles maintain a high level of service throughout their connections.

   - **Distance-based QoS**

     Distance-based QoS is 100% when the vehicle is within the RSU's range. If the vehicle is outside this range, the QoS decreases exponentially as the distance increases. This can be expressed mathematically as:

$$\text{QoS}_{v_i,\text{dist}} = \begin{cases} 1 & \text{if } d_i \leq R_j \\ e^{-\alpha(d_i - R_j)} & \text{if } d_i > R_j \end{cases}$$

where d_i is the distance between vehicle i and RSU j, R_j is the coverage radius of RSU j, and α is a decay constant.

This exponential decay model effectively captures how distance from an RSU impacts the QoS experienced by a vehicle. In wireless networks, signal strength, and therefore QoS, naturally decreases as the distance from the signal source increases. By mapping distance to QoS, this model provides a simplified yet accurate representation of this relationship, reflecting real-world behaviors such as signal attenuation and increasing latency as vehicles move away from the RSU.

While distance is a static measure, it is a reliable proxy for these dynamic factors, making it a helpful model for predicting and managing network performance in a vehicular environment. In more complex scenarios, this distance-based QoS would correspond to real measures like Signal-to-Noise Ratio (SNR) and connection stability, making it a practical abstraction for simulations that optimize network strategies. [24, 26, 35]

- **Load-based QoS**

  Load-based QoS is 100% if the RSU's load is within its capacity. Note that the RSU's provided QoS directly affects the QoS of all connected vehicles. If the load exceeds capacity, QoS deteriorates proportionally, calculated as:

$$\text{QoS}_{\text{rsu}_j,\text{load}} = \begin{cases} 1 & \text{if load}_j \leq \text{capacity}_j \\ \frac{\text{capacity}_j}{\text{load}_j} & \text{if load}_j > \text{capacity}_j \end{cases}$$

  This model captures how QoS declines when an RSU is overloaded. In a real-world scenario, an overburdened RSU would likely drop or delay some tasks, leading to a decrease in QoS for all connected vehicles. The proportional reduction in QoS in this model reflects how resources become strained, resulting in lower service quality.

  By simulating these conditions, the load-based QoS model effectively mirrors real-world issues like increased latency and packet loss during network congestion. This helps in testing strategies to manage and mitigate the impact of overloading, ensuring that service quality is maintained even under high demand. [1, 17, 24]

- **Total QoS**

  To combine these two QoS metrics, a multiplication approach is preferred as it more accurately reflects scenarios where a significant drop in either distance-based or load-based QoS severely impacts overall service quality. This combined QoS model captures the impact of both physical distance and the RSU's current load on the vehicle's experience.

The combined QoS can be calculated as:

$$\text{QoS}_{v_i} = \text{QoS}_{v_i,\text{dist}} \times \text{QoS}_{\text{rsu}_j,\text{load}}$$

where rsu_j is the RSU currently connected to vehicle v_i.

This approach ensures that if either the signal strength (affected by distance and range) or the network load (affected by load and capacity) deteriorates, the overall QoS for the vehicle will reflect the resulting degradation in service, providing a realistic assessment of the network's performance.

4. **Equality of Load Distribution**

   The equality of load distribution across RSUs is measured using the Gini coefficient, a widely used metric for assessing inequality. A Gini coefficient of 0 indicates perfect equality, while a value of 1 indicates maximum inequality. It is mathematically expressed as:

$$G = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |\text{load}_i - \text{load}_j|}{2n^2\mu}$$

   where G is the Gini coefficient, n is the number of RSU, load_i and load_j are the loads of the RSU i and j respectively, and μ is the mean load. In this context, the load on each RSU includes any load beyond its capacity, allowing overloads to influence the results.

   By including excess load in the Gini coefficient calculation, we capture both the distribution of load and the strain on the network when certain RSUs are overloaded. This approach provides a more comprehensive measure of inequality, reflecting not only the fairness of load distribution but also the inefficiencies that arise under stress.

   The Gini coefficient effectively evaluates load distribution strategies in vehicular networks. Ensuring even load distribution across RSUs optimizes resource utilization, prevents bottlenecks, and enhances overall network performance. By accounting for load beyond capacity, we gain deeper insights into the network's behavior under stress, particularly in scenarios where load inequality may lead to RSU overload and QoS degradation.

Benchmarks

To assess the performance of our proposed ARHC handover strategy, we will compare it against several benchmarks:

1. **Proposed ARHC Strategy**
   This strategy is the novel agent-based handover coordination developed in this thesis, leveraging MAS at the edge and fog layers for decentralized coordination.

2. **MAS Handover Strategy from Offloading Work (Baseline)**
   This benchmark involves a strategy where vehicles always connect to the nearest RSU, as described by Grislin-Le Strugeon et al. [13] and Ouarnoughi et al. [33]. It serves as a baseline to understand the benefits of more sophisticated handover mechanisms.

3. **Earliest/Latest Possible Handover (Baseline)**
   These benchmarks involve handing over at the earliest or latest possible moment based on the vehicle's distance and the RSU's range. They serve as control benchmarks to measure the performance of more advanced strategies and provide insights into the advantages of timely handover decisions.

Evaluation Goals

The primary goals for evaluating the ARHC handover strategy are as follows:

1. **Minimize Successful Handovers**
   Reducing the number of successful handovers helps to minimize the state migration overhead and associated communication costs. Efficient handover strategies aim to achieve necessary handovers with the least frequency.

2. **Minimize Failed Handovers**
   Ensuring that handover attempts are efficient and effective is crucial. Fewer failed handovers indicate that the handover decision-making process is robust and reduces unnecessary communication overhead.

3. **Maximize Average and Minimum QoS**
   Maintaining a high QoS for all vehicles is a priority. Focusing on both the average and minimum QoS ensures that the overall network performance is optimized while avoiding service degradation for individual vehicles.

4. **Minimize Inequality in Load Balancing**
   Striving for an even distribution of computational load across RSUs helps optimize resource utilization and enhance overall network performance. Minimizing inequality in load balancing ensures that no single RSU is overwhelmed while others are underutilized.

By systematically collecting and analyzing these metrics, we can comprehensively evaluate the effectiveness of the proposed handover strategies and compare them against established benchmarks. This structured approach will highlight strengths and potential improvement areas, guiding future enhancements in VEC handover coordination.

4.7.2 Experimental Setup

This section outlines the experimental setup used to evaluate the handover coordination strategies in VEC environments.

Simulation Environment Configuration

The simulation environment is built using the Python Mesa framework [20], providing a flexible agent-based modeling platform. Key components include:

1. **Vehicular Network Topology**
   A realistic network with multiple RSUs and vehicles, with RSUs ensuring overlapping coverage and vehicles following real-world mobility patterns derived from urban datasets like the Créteil roundabout dataset [22].

2. **Agent-based System**
   MASs are used for decentralized handover coordination, with RS agents handling communication and handovers and VE agents assisting in decision-making.

3. **Communication Protocols**
   Standard message exchange protocols include CAMs from vehicles and basic load information from RSUs.

Simulation Parameters

The simulation parameters reflect realistic operational conditions:

1. **RSU Agent Parameters**
   - Frequency of load information exchange: Configurable (e.g., every 1s, 5s, 10s).
   - RSU capacity: Varied to simulate different load handling capabilities.

2. **Vehicle Parameters**
   - Frequency of CAM publication: Fixed at 1 Hz.
   - Mobility model: Based on real-world urban data traces.

3. **Simulation Duration**
   Each simulation run lasts several minutes to multiple hours, depending on the dataset used, to capture transient and steady-state behaviors.

By setting up this detailed and realistic simulation environment focused on urban scenarios, we aim to thoroughly evaluate the proposed ARHC strategy in VEC environments, providing insights into their performance and scalability.

4.7.3 Expected Outcomes

The expected outcomes from the simulation aim to comprehensively evaluate ARHC. These outcomes will highlight the effectiveness of the strategies in various metrics and scenarios, contributing significantly to the understanding and improvement of VEC environments.

1. **Comparison of Strategies**
    - We expect our proposed ARHC strategy to outperform the nearest-RSU strategy and the baseline approaches (earliest and latest handover). This is particularly evident in minimizing the number of successful handovers while maintaining high QoS. The ARHC strategy is anticipated to guarantee a consistently high QoS, ideally close to 100%, across various scenarios.
    - The ARHC strategy should also result in a more balanced load distribution across RSUs and a lower number of failed handovers compared to the other strategies.
2. **Communication Efficiency**
    - We anticipate that decreasing the frequency of communication between RSUs will increase the total number of handovers, including successful and failed attempts. This is expected to negatively impact the QoS, particularly the minimum QoS experienced by any vehicle.
    - Efficient communication patterns are crucial for improving the overall effectiveness of the handover process. ARHC should balance the number and size of messages exchanged to optimize communication efficiency.
3. **Simulation Suitability**
    - Our simulation environment is expected to be highly suitable for evaluating MASs strategies for handover coordination in vehicular networks. The detailed and dynamic nature of the simulation will provide valuable insights into the practical implications of different handover strategies.
    - The simulation will help identify the strengths and weaknesses of each strategy, guiding future improvements and optimizations in VEC handover coordination.

The expected outcomes from the simulation will clearly indicate the effectiveness of the proposed ARHC strategy in terms of QoS, load distribution, and communication efficiency. By systematically comparing ARHC with existing methods and baseline approaches, we aim to demonstrate its superiority in maintaining high service quality and efficient resource utilization. The insights gained from the simulation will be crucial for advancing the state of VEC handover coordination, ultimately enhancing the performance and reliability of vehicular networks.

4.8 Conclusion of the Approach

This chapter introduced the ARHC strategy for VEC environments, emphasizing the integration of MASs within RSU-based systems. Our approach, the ARHC strategy, addresses the challenges of seamless session relocation across multiple MEC/VEC servers, particularly for computationally intensive AD tasks.

We explored and compared centralized, vehicle-based, and RSU-based strategies, underscoring the advantages of decentralized decision-making at the fog layer. This method enhances system performance by balancing computational loads, reducing unnecessary handovers, and minimizing state migration overhead.

The chapter also detailed the network model and agent architecture, focusing on the interaction between VE and RS agents. Simplifications, such as assuming 100% offloading of tasks to RSUs, were made to keep the focus on the primary objectives of the thesis.

The following chapters will explore implementing the agent models and the simulation environment developed to evaluate ARHC. This includes the setup, scenarios, data collection methods, and evaluation strategies. The final chapter will comprehensively assess the handover coordination strategies, providing insights into their effectiveness and practical implications.

CHAPTER 5

Implementation

This chapter provides an in-depth description of the implementation of the agent model and the simulation environment, both central to evaluating the proposed ARHC strategy within VEC environments. The ARHC strategy is the result of the coordinated behaviors of the In-Vehicle (VE) and RoadSide (RS) agents, each performing distinct but complementary roles. As outlined in the approach chapter, Figures 4.2 and 4.3 illustrate the key components of these agents and their communication model. These figures serve as the foundation for understanding the detailed implementation processes discussed in this chapter.

The chapter is divided into three main sections: The implementation of VE agents, the implementation of RS agents, and the custom simulation environment developed to evaluate the ARHC strategy.

5.1 In-Vehicle (VE) Agent Implementation

The VE agent is responsible for continuously monitoring the vehicle's status, sensing environmental conditions, and communicating relevant data to the RS agents. The key functionalities of the VE agent and their implementation details are outlined below.

5.1.1 Monitoring System Activities and Environment

The VE agent tracks various parameters to determine the vehicle's dynamic load at any given time t. These parameters include:

- **Position Tracking**: Utilizes GPS data to update the vehicle's location periodically. In the simulation, this is modeled on a 2D grid representing the environment, with positions updated at regular intervals.

- **Speed**: Measures the vehicle's speed to aid in load determination and future position prediction.
- **Nearby Vehicles**: Considers the number of vehicles in proximity, impacting computational load.
- **Random Factor**: Introduces an element of randomness to account for unpredictable variables. This factor is included to simulate real-world unpredictability in computational load.

5.1.2 Communication

The VE agent generates CAMs at a frequency of 1 Hz, as required by the ETSI EN 302 637-2 standard [10]. These messages include:

- **Position Data**: Current GPS coordinates of the vehicle.
- **Speed and Direction**: Real-time speed and heading of the vehicle.

Although CAM dissemination is not simulated, the system ensures that the necessary data is always available to RS agents. This is based on the assumption that a minimum publish rate of 1 Hz is consistently met.

5.1.3 Supervision and Task Offloading

The VE agent's primary responsibilities include monitoring its status, updating it, and playing a role in task offloading. This involves:

- **Supervision of Offloading**: Monitoring the status of offloaded tasks to ensure proper processing by RSUs. While this is crucial in a real-world setting, it is not this thesis's primary focus and is simplified in the current model. This aspect is more thoroughly explored in works such as Grislin-Le Strugeon et al. [13], Ouarnoughi et al. [33], and Ahmed et al. [1].
- **Communication Protocols**: Implementing robust protocols for reliable communication with RSUs for task offloading. This is assumed to function correctly within the simulation, following the assumptions made in the approach section.

5.1.4 Data Collection

The VE agent-related metrics are collected at the end of each timestep of the simulation. Each VE agent computes its currently experienced QoS based on its distance and the current QoS offered by its respective RSU, determined by the RSU's load and capacity. For more details on how these values are computed, refer to Section 4.7.1.

In the context of the ARHC strategy, the VE agents play a crucial role in monitoring the vehicle's status and communicating vital information to the RS agents. Their continuous feedback and task-offloading decisions are integral to maintaining seamless handovers and optimal resource utilization, which are central to the success of the ARHC strategy.

5.2 Roadside (RS) Agent Implementation

The RS agent is pivotal in handover coordination and task offloading. The implementation can be divided into three primary phases: Monitoring, Handover Decision-Making, and Handover Execution. Each phase comprises particular components and strategies designed to ensure efficient and effective handover processes, contributing to the overall performance and reliability of the ARHC strategy in the VEC environment.

Monitoring Phase

In this phase, RS agents continuously gather and process information about the vehicles and their own load status. This data is crucial for making informed decisions during the handover process, which is central to the ARHC strategy. Key activities include:

- **CAMs**: The RS agent receives and processes CAMs from VE agents to provide real-time updates on vehicle positions, speeds, and directions. For simplicity, the simulation assumes direct access to vehicle locations through an intermediary data plane, controlling CAM data available to RSUs.
- **Current Load Calculation/Observation**: The RS agent continuously calculates the current load at timestep t by summing the computational requirements of all connected vehicles at timestep t.
- **Load Sharing and Neighboring RSUs**: RS agents periodically exchange load information with neighboring RS agents to enable collaborative decision-making and load balancing. This communication enables collaborative decision-making and load balancing, helping to predict future load conditions and enhance the overall handover decision-making process within the ARHC strategy. When performing handovers, both the source and target RS agents update their information about each other's load to ensure synchronized load awareness.

5.2.1 Handover Decision-Making Phase

Handover decision-making in the ARHC strategy involves determining if handovers are necessary, defining explicit triggers, and selecting vehicles and target RSUs for handover. This phase can be divided into the following components:

Decision-Making Heuristics

The decision-making process for handovers relies on a set of heuristics that evaluate the suitability of a target RSU for each vehicle. These heuristics consider both distance-related and load-related factors:

1. **Location-Related RSU Suitability**

   This suitability heuristics is used to determine the suitability of an RSU for a vehicle based on their locations and the vehicle's trajectory. For simplification, the vehicle's trajectory is approximated by a combination of the vehicle's position and orientation relative to the RSU's position.

   With our metric, we try to gain a single value that is indicative of 1. the distance between the vehicle and RSU relative to the RSU's range, and 2. the orientation relative to the direction from the vehicle to the RSU, to determine if the vehicle is moving towards, away, or orthogonal to the RSU. The criteria should be low if one is met and approach zero if both are met, i.e., the vehicle will leave the station's range soon.

   The formula for our location-based station suitability heuristic is the following:

$$S_{\text{loc}} = \begin{cases} 1 - \frac{0.5*\cos(\theta)+0.75}{1.25} * \frac{d_i}{R_j}, & \text{if } d_i \leq R_j \\ 0, & \text{if } d_i > R_j \end{cases}$$

   where θ is the angle between the vehicle's orientation and the direction from the vehicle to the RSU, and d_i is the current distance between the vehicle i and RSU j with range R_j.

   The heuristic is designed to evaluate the following criteria:

   - Regarding the value range, $0 \leq S_{\text{loc}} \leq 1$ holds as long as the vehicle is in range of the RSU.
   - If $d = 0$ (the vehicle and the RSU share the same location), then $S_{\text{loc}} = 1$.
   - If $d = d_{\text{max}}$ (the vehicle is at the RSU's range) and $\theta = \pm\pi = \pm 180°$ (the vehicle is driving away from the station), then $S_{\text{loc}} = 0$.
   - The suitability decreases linearly as the vehicle moves directly toward the station and increases linearly if the vehicle moves directly away from the station.

   The constants were determined experimentally, ensuring the suitability score reflects the vehicle's likelihood of staying connected to the RSU.

2. **Load-Related RSU Suitability**

   The load-related RSU suitability is simpler to calculate than the location-related suitability. It can be determined as follows:

$$S_{\text{load}} = \begin{cases} \frac{C_j - (L_j + L_i)}{C_j}, & \text{if } L_j + L_i \leq C_j \\ 0, & \text{if } L_j + L_i > C_j \end{cases}$$

where C_j stands for RSU j's capacity, L_j for j's current load, and L_i for the load currently offloaded by vehicle i.

Key observations:

- Similar to S_{loc}, the suitability score is bounded by $S_{\text{load}} \leq 1$. However, if the combined load exceeds the RSU's capacity C_j, S_{load} is set to 0.
- The suitability score S_{load} considers both the RSU's current load and the additional load from the vehicle i post-handover. This calculation reflects the RSU's available capacity relative to its total capacity after accounting for the potential handover.

3. **General RSU Suitability**

   To determine the general suitability of an RSU, the two specific suitability scores are combined:

$$S = S_{\text{loc}} \cdot S_{\text{load}}$$

   This approach ensures that both the vehicle's trajectory and the RSU's capacity are considered, helping to identify the most appropriate RSU for a handover within the ARHC strategy.

Handover Triggers

Triggers for handovers can be categorized either as imperative or alternative, as described in Section 2.2.3.

1. **Imperative Trigger**:
   - **Vehicle Leaving Range**: Initiated when a vehicle is near the RSU's maximum range and moving away from it. This is required to keep the distance-based QoS high. Using the heuristics introduced earlier in this section, we can use a threshold on S_{loc} to determine if a location-related imperative handover is necessary.
   - **RSU Overload**: This condition occurs when the load on an RSU approaches or exceeds its capacity. To manage this, the RS agent must determine which subset of connected vehicles should be handed over to neighboring RSUs. This decision constitutes a MCDM problem involving factors such as the vehicles' positions and trajectories, the current load on the RSU, and the load status of adjacent RSUs. An overload-induced handover is triggered when the RSU

load surpasses a threshold of 0.7, as detailed in Section 6.1.4. To prevent unnecessary handovers to RSUs with higher loads, a load-balancing hysteresis of 0.05 is applied.

2. **Alternative Trigger**:

   - **Load Balancing**: This mechanism is activated when the load status of neighboring RSUs indicates a significant imbalance. If a neighboring RSU has a substantially lower load and vehicles are within its range and optimally moving towards it, an alternative handover is initiated to enhance overall system efficiency and balance the load across the RSUs within the ARHC strategy. To prevent shifting vehicles to RSUs that might already be under strain, a load-balancing hysteresis of 0.05 is applied. A minimum load-balancing suitability threshold of 0.3 is set, ensuring that only vehicles meeting this suitability criterion with the target RSU are considered for handover by the source RSU. More details on the load-balancing suitability criteria can be found in Section 5.2.1. Further information on the ARHC strategy parameters are provided in Section 6.1.4

During the initial development of the agent model, a challenge known as the "ping-pong effect" emerged, where vehicles frequently switched between two RSUs with similar loads. This issue, common in cellular handovers, [34] was mitigated by the mechanisms discussed earlier, including using hysteresis, suitability thresholds, and specific trigger thresholds.

These triggers, informed by the decision-making heuristics, guide the RS agent in executing handovers that maintain QoS and optimize resource utilization across the network.

Decision-Making Algorithm

The heuristics and suitability scores previously described drive the decision-making algorithm for handovers in the ARHC strategy. The process varies depending on the trigger type:

1. **Vehicle Leaving Range**:

   - When a vehicle is nearing the edge of the current RSU's range, the algorithm focuses on that specific vehicle. It calculates the suitability scores for all neighboring RSUs within range.
   - The neighboring RSUs are then traversed in order of decreasing suitability. The vehicle is handed over to the first RSU that accepts the handover request, ensuring continued connectivity without interruption.

2. **Load Balancing and Overload**:

   - For scenarios involving load balancing or RSU overload, the algorithm considers all connected vehicles. It computes the suitability scores across the cross-product of these vehicles and all neighboring RSUs within range.
   - This list of potential handovers is sorted by suitability, and handovers are executed sequentially, starting with the highest scores. The process continues until the load on the RSU is reduced below the critical threshold. This evaluation and potential rebalancing are performed at fixed intervals, typically every second, to maintain optimal load distribution for regular load balancing.

This algorithm ensures that handovers are performed efficiently and effectively, minimizing disruptions and optimizing resource utilization across the VEC environment.

5.2.2 Handover Execution Phase

The Handover Execution phase involves the steps the RS agent takes to carry out a handover once it has been decided.

Handover Request: When a handover is necessary, the initiating RS agent sends a request to the target RS agent. For simplicity in this thesis, it is assumed that:

- Alternative handover requests are accepted if the target RS agent has sufficient capacity and the vehicle is in range.
- Imperative handovers, marked as urgent, are always accepted by the target agent.

Vehicle Handover: Once the target RS agent accepts the request, the source RS agent completes the handover by transferring the vehicle's connection and computational tasks to the new RSU. The simulation updates the vehicle's associated RSU and registers it with the target agent. The actual connection transition is abstracted for simplicity.

State migration: The RS agent also manages the migration of the vehicle's computation state, including any offloaded tasks stored on the current RSU. This ensures continuity of service during the handover. In this thesis, state migration is simplified: It is assumed that all states are of equal size, and the number of state migrations corresponds directly to the number of successful handovers, as described in Section 4.4.2.

5.2.3 Integration into the ARHC Strategy

The ARHC strategy is driven by the coordinated actions of VE and RS agents. VE agents monitor vehicle status and provide real-time updates, while RS agents use this data to manage handovers and balance network loads. This collaboration ensures the network remains adaptive and maintains high QoS with minimal disruptions.

By integrating these agents' capabilities, the ARHC strategy effectively minimizes unnecessary handovers, optimizes resource utilization, and ensures consistent service quality across the vehicular network.

The next sections will explore the custom simulation environment created to evaluate the ARHC strategy under various real-world conditions.

5.3 Simulation Implementation

This section details the implementation of the simulation environment, which is designed to evaluate the effectiveness of agent-based handover coordination strategies in VEC. The simulation is critical for testing the ARHC strategy under realistic vehicular conditions, focusing on urban scenarios where seamless handover and load balancing are essential.

5.3.1 Simulation Environment Setup

This section details the simulation environment built with Python Mesa, focusing on time simulation, task execution, scheduling, and performance optimizations.

Platform Choice: Python Mesa

The simulation environment is built using the Python Mesa framework, which is well-suited for agent-based modeling. Mesa provides a flexible and extensible platform, enabling efficient simulation and visualization of agent behaviors over time. The choice of Mesa is driven by its capabilities similar to other well-established MAS simulation frameworks like NetLogo, Repast, and MASON. Mesa's built-in tools for visualizing and analyzing agent interactions are particularly useful for this study. [20]

Simulation Mechanics

The simulation operates in discrete timesteps of 1 second. This interval was chosen because it aligns with the worst-case update interval for CAMs, typically transmitted at 1-10 Hz. This frequency ensures that the simulation accurately reflects the timing of vehicle status updates in real-world scenarios.

Moreover, the 1-second interval is sufficient given the nature of relatively infrequent handover events. Even under optimal conditions, a vehicle is not expected to undergo a handover within fractions of a second; with an effective strategy, handovers should be spaced out over several seconds. This interval balances capturing essential vehicle dynamics and maintaining computational efficiency, allowing for a realistic vehicular movement and communication simulation.

The tasks represent AD computations offloaded to fog infrastructure, such as neural network-based vision or path planning. The computational load of these tasks is quantified in terms of Floating Point Operations (FLOP), reflecting the processing requirements. For simplification, only general FLOP requirements are considered rather than specifying

a list of specific tasks with distinct FLOP needs, as described in Section 4.4.2. The processing capabilities of RSUs are given as Floating Point Operations Per Second (FLOPS), also referred to as the RSUs' capacities.

Scheduling and Strategy Pattern

The simulation follows a structured schedule where VE agents update their state (e.g., location, speed) first, followed by RS agents executing their coordination strategies. Each timestep involves shuffling the execution order of agents using a seeded randomizer to ensure deterministic results and comparability across simulation runs.

Additionally, shared information on other RSUs' current loads is collected before each step, according to the load information sharing interval set by the scenario configuration. Therefore, all RSUs have the same initial knowledge of the system after each load-sharing cycle, independent of their execution order for the following step.

The RS agents utilize a strategy pattern to enable the implementation of various coordination policies. This pattern provides methods for initializing RSU-specific fields and executing the main logic for each timestep, allowing flexible and extensible strategy development. This also serves as the primary point for researchers interested in designing and evaluating their strategies or modifying existing ones.

Performance Optimizations

While Python may not offer the fastest execution times, several optimizations are implemented to enhance performance. These include executing independent simulation runs in parallel and dynamically adding or removing vehicles from the simulation based on their presence in the area of interest, reducing unnecessary computations.

5.3.2 Scenario Development

The scenarios used in the simulation are derived from real-world datasets to ensure realistic conditions. This section details the development and configuration of scenarios using these datasets.

Créteil Roundabout Dataset

The Créteil Roundabout Dataset, presented in [22], provides detailed traffic data for the Europarc roundabout in Créteil, France. This dataset is crucial for evaluating local decision-making in ITS and supports hybrid decentralized ITS by providing detailed micro-mobility traces, making it highly relevant for testing the ARHC strategy in dense urban environments. [1] Its relevance is further highlighted by its use in studies like Rejiba et al. [35], where it was employed to simulate VEC scenarios, demonstrating its

[1] The dataset is available for download at `https://vehicular-mobility-trace.github.io/`

effectiveness in testing network performance and decision-making strategies in dynamic VEC environments.

Dataset Characteristics

- The roundabout has six entrances/exits, roads with 2-3 lanes, one bus lane, four changing-lane spots, and 15 traffic lights.
- It includes approximately 10,000 trips during peak hours, covering two distinct periods: morning (7 AM - 9 AM) and evening (5 PM - 7 PM).
- While the dataset distinguishes between vehicles and buses, this distinction is not relevant to the scope of this thesis and is therefore not considered in the analysis.

Data Preprocessing

- **Area Reduction**: The dataset was initially extensive, covering areas beyond the roundabout. For the simulation, it was reduced to focus only on the central roundabout and its immediate intersections, eliminating irrelevant data points.
- **Coordinate System Adjustment**: Orientations and coordinates were adjusted to fit the simulation's coordinate system, ensuring an accurate representation of vehicle movements.
- **Road Tile Extrapolation**: For visualization, road tiles were plotted on a binary 2D grid by marking tiles where vehicles traveled based on distinct integer positions. This method helps visualize the simulation environment's road network and traffic flow.

Current Usage

Both the morning (7-9 AM) and evening (5-7 PM) datasets are employed in the simulation to provide a comprehensive evaluation of the ARHC strategy under different traffic conditions. This approach enables a detailed analysis of traffic dynamics during peak periods, as described later in Section 6.1.3.

Configuration Options

The simulation is highly configurable, allowing for extensive scenario customization and adaptability. Key configuration options include:

1. **Strategy Configuration**
   - **Strategy Selection**: Different handover coordination strategies can be selected, including the ARHC strategy with specific parameters.

- **Strategy Parameters**: Optional parameters such as load sharing interval can be configured to fine-tune the strategy's performance.

2. **RSU Configuration**

   - **Number of RSUs**: The number of RSUs in the simulation can be adjusted to study the impact of different infrastructure densities.
   - **Placement**: RSUs can be placed at specific locations within the simulation environment to mimic real-world setups.
   - **Range**: The communication range of each RSU can be configured to evaluate different coverage scenarios.
   - **Capacity**: The computational capacity of each RSU can be adjusted to test various load-handling capabilities.

3. **Vehicle Load Generation**

   - **Load Generation**: Although fixed for all experiments, the load generation parameters are described in further detail in the evaluation section, where a dynamic distribution of computational loads is implemented based on real-world AD tasks as specified by Ouarnoughi et al. [33].

These extensive configuration options allow a comprehensive analysis of the baseline strategies and ARHC under different conditions and infrastructural setups.

5.3.3 Data Collection and Analysis

This section describes the methods for collecting and analyzing data during simulation runs. The focus is on the exact implementation details, ensuring the collected data is accurate and efficiently processed for analysis.

Data Export

A robust data export mechanism is implemented to facilitate the analysis of simulation results. The simulation environment is prepared to support batch execution and export data to CSV files, ensuring that results are accurately captured and easily accessible for post-processing. This automated export process minimizes the risk of errors from manual data handling and streamlines the analysis workflow.

Visualization

The simulation environment incorporates integrated visualization tools to provide real-time insights into the simulation process. Browser-based visualization components, embedded within Jupyter Notebooks, offer a dynamic view of the agent grid, road layout, vehicle positions, RSU positions, and their respective ranges. Custom visualizations, such

as line charts displaying recent data trends, are also included to aid in developing and analyzing agent models.

These visualizations serve multiple purposes:

- **Real-time Monitoring**: Allowing the researcher to observe the simulation as it progresses, ensuring that all components function as expected.
- **Development Support**: Helping debug and refine agent behaviors by visually representing their actions and interactions.
- **Analysis Aid**: Offering visual summaries of key metrics, such as RSU load and vehicle handovers, to facilitate a deeper understanding of the simulation outcomes.

Metrics Collection

Metrics related to VE and RS agents are collected at the end of each timestep, as described in 4.7.1. These metrics include both direct measures, such as load and QoS, and derived metrics, which provide deeper insights into system performance:

- **VE Agent Metrics**: These include the current QoS experienced by the vehicle, calculated based on its distance, the RSU's range, and the QoS provided by the connected RSU, which in turn is determined by the RSU's load and capacity.
- **RS Agent Metrics**: Metrics include the current load, the number of connected vehicles, and the number of successful handovers.

Additionally, derived metrics are calculated from these basic metrics to provide deeper insights:

- **Average QoS**: This captures the overall QoS experienced by vehicles, calculated as the mean of all individual vehicle QoS values at each timestamp.
- **Minimum QoS**: This reflects the lowest QoS experienced by any vehicle at each timestamp, providing a measure of the worst-case performance in the network.
- **Handover Frequency**: This metric tracks the number of handovers per second, indicating the efficiency of the handover process and identifying periods of high activity.
- **Load Distribution Gini Coefficient**: This measure reflects the evenness of the load distribution among all active RSUs, providing insights into load balancing within the network.

These metrics are essential for evaluating the performance of the handover coordination strategies.

Analysis Methods

The collected data is analyzed using statistical methods and visualization techniques to extract meaningful insights. Key analysis steps include:

- **Data Cleaning**: Ensuring that the exported data is free from errors and inconsistencies.
- **Descriptive Statistics**: Summarizing the data to provide an overview of the simulation results, including mean, median, and standard deviation of key metrics.
- **Comparative Analysis**: Comparing the performance of different strategies and configurations to identify the most effective approaches.
- **Trend Analysis**: Identifying patterns and trends over time, such as the impact of varying RSU capacities or vehicle loads on handover frequency and QoS.

By implementing these rigorous data collection and analysis methods, the simulation offers a comprehensive framework for evaluating the effectiveness of the ARHC strategy in agent-based handover coordination within VEC environments. The following evaluation will test ARHC's ability to manage handovers, maintain high QoS, and balance loads, offering insights into its practical application in real-world scenarios.

CHAPTER 6

Evaluation

The detailed simulation setup forms a solid foundation for evaluating the Agent-based RSU Handover Coordination (ARHC) strategy in VEC environments. In the upcoming evaluation chapter, simulations will test this framework to assess ARHC's performance in managing handovers, maintaining high QoS, and balancing computational loads. By comparing sparse and dense RSU configurations, the evaluation will provide key insights into the strategy's scalability and real-world applicability.

6.1 Experimental Setup

This section outlines the setup for evaluating the ARHC strategy in a VEC environment. Table 6.1 summarizes the key simulation parameters, which provides an overview of the various configurations and settings used throughout the experiments.

6.1.1 Load and Capacity Model

This subsection defines the computational and infrastructural parameters for the simulation, establishing the foundation for evaluating the ARHC strategy.

Task Computation Model

The computational tasks and their respective loads are derived from the work of Ouarnoughi et al., who provide a detailed breakdown of the computational requirements for typical autonomous driving tasks. To enhance real-world applicability, we focus on computational loads associated with AD tasks, which are critical in VEC environments. Detailed task descriptions are available in the implementation of the task entity module, which can be accessed online. [1] [33] These tasks are summarized in Table 6.2.

[1] The task entity implementation is available at `https://github.com/Houarnoughi/Simpy-AD/blob/develop/simulation/entity/task.py`, last accessed on August 3, 2024.

Table 6.1: Simulation Parameters for Evaluation

Parameter	Value / Description
Scenario	Créteil Roundabout, Morning and Evening Traces
Simulation Area	200x200 meters
Number of Vehicles	Morning: 3929 traces, max 92 concurrent vehicles Evening: 3881 traces, max 74 concurrent vehicles
RSU Configurations	Sparse: 4 RSUs Dense: 9 RSUs
RSU Range	70 meters
RSU Capacity	Full: 65 TFLOPS Half: 32.5 TFLOPS Quarter: 16.25 TFLOPS
Vehicle Capacity	NVIDIA TX2, 1.3 TFLOPS, 70% utilized (0.91 TFLOPS)
Task Computation Load	79.72 GFLOP per frame (2.391 TFLOPS at 30 FPS)
Dynamic Load Distribution	Uniform distribution between 1.9 and 3 TFLOPS, avg. 2.45 TFLOPS
Load Sharing Frequency	1 second (for ARHC strategy tuning)
Traffic Data Timeframe	Morning: 7:15 AM - 8:45 AM Evening: 5:15 PM - 6:45 PM
RS Agent Parameters	Leaving Threshold: 0 Overload Threshold: 0.7 Load-Balancing Hysteresis: 0.05 Minimum Suitability: 0.3

The total computational load per vehicle at a frame rate of 30 FPS is approximately 79.72 GFLOP per frame, equating to 2.391 TFLOPS. This frame rate is consistent with the studies by Ouarnoughi et al. and Le-Strugeon et al., highlighting the importance of maintaining high computational throughput for AD applications. [13, 33]

Vehicle Onboard Capacity

Each vehicle is equipped with an NVIDIA TX2 processor, capable of delivering up to 1.3 TFLOPS. [2] In our model, we assume that each OBU can handle up to 70% of this capacity (0.91 TFLOPS) locally, with the remaining load offloading to the RSUs. This approach aligns with standard practices in VEC research, as noted by Ouarnoughi et al. and Grislin-Le Strugeon et al. [13, 33]. Although they suggest a more dynamic interaction where vehicles and RSUs communicate to decide on the load to offload, we simplify this by assuming that any load exceeding 70% of the vehicle's capacity is offloaded. This simplification streamlines our model for this thesis.

[2] For more information, refer to NVIDIA's official documentation on Jetson modules at `https://developer.nvidia.com/embedded/jetson-modules`, last accessed on August 3, 2024.

Table 6.2: Computational Load for AD Tasks

Task	**Load (GFLOP)**
Traffic Sign Detection	23
Lane Detection	0.23
Object Detection	0.23
Object Tracking	0.23
Mapping	0.23
Localization Algorithm	0.23
Motion Prediction	0.23
Trajectory Planning	2.3
Behavior Planning	2
Route Planning	50
Control Algorithm	1
Traffic Light Detection	0.04
Total Load per Frame	**79.72**

RSU Capacity and Configuration

The RSUs are equipped with NVIDIA Tesla T4 Graphics Processing Units (GPUs), providing a peak performance of 65 TFLOPS [3] This capacity is chosen to handle the aggregated computational loads from multiple vehicles efficiently. The GPU capacity can be scaled for experimental scenarios requiring varied computational capacities. For example, halving the capacity for a particular experiment would adjust the Tesla T4 capacity to 32.5 TFLOPS.

Details on the RSU capacity for each experiment are provided in the scenario-specific sections, which consider factors such as the mobility dataset, RSU density, and the experiment's aim.

Dynamic Load Distribution

To introduce more dynamism into the simulation, the load generated by a vehicle is distributed over a uniform distribution of 1.9 to 3 TFLOPS, approximating a ±25% variation. This reflects realistic scenarios where computational demands fluctuate due to varying environmental conditions and task complexities. Each vehicle produces an average of 1.54 TFLOPS that are offloaded to RSUs. A seeded random generator ensures deterministic and reproducible results.

[3]Further details on the NVIDIA Tesla T4 GPU can be found at `https://www.nvidia.com/en-us/data-center/tesla-t4/`, last accessed on August 3, 2024.

6.1.2 RSU Parametrization

RSU Range

In the literature, the range of RSUs typically falls between 150 and 300 meters, influenced by several additional factors, such as memory bandwidth and wireless network congestion, which impact the effective range and capacity of RSUs. However, for the relatively small scale of the proposed simulation scenarios (e.g., the relevant section of the Créteil roundabout dataset spans 200x200 meters), we limit the RSU range in our simulations to $d_0 = 70$ meters to ensure practical applicability and model simplicity. [22, 24, 25, 26, 35]

This choice ensures that approximately four RSUs can effectively cover the simulation area in a sparse distribution, with the flexibility to add more for denser distributions. The decision is based on empirical data and typical RSU specifications, which suggest that shorter ranges are more realistic and practical for urban scenarios with numerous obstacles. [24, 25, 26, 35]

RSU Capacity

As detailed in Section 6.1.1, the RSUs are equipped with NVIDIA Tesla T4 GPUs, providing a peak performance of 65 TFLOPS. This capacity was selected to efficiently handle the aggregated computational loads from multiple vehicles, with the option to adjust RSU capacity for different experimental scenarios.

Distance-based QoS

An exponential decay function is used to model the QoS experienced when vehicles move outside the optimal range of an RSU, as described in Section 4.7.1.

Empirical observations and simulation results are considered to determine the decay constant α. For instance, if at 100 meters the QoS drops to 50% of its maximum value, α can be solved as:

$$\alpha = -\frac{\ln(0.5)}{100 - 70} \approx 0.0231$$

Studies by Huang et al. and Lin et al. examine the impact of vehicles moving beyond the optimal range of RSUs, revealing a significant drop in signal strength and a rise in interference, which together result in an exponential decline in QoS metrics. These findings robustly support the application of an exponential decay model and validate the selected parameters for their research. [17, 24]

This comprehensive approach ensures that our model accurately represents real-world conditions and provides a robust framework for designing and managing vehicular networks. The exponential distance-based QoS model is implemented as follows:

$$\text{QoS}_{v_i,\text{dist}} = \begin{cases} 1 & \text{if } d_i \leq 70 \\ e^{-\alpha(d_i-70)} & \text{if } d_i > 70 \end{cases}$$

where d_i is the distance between vehicle i and RSU j, and $\alpha \approx 0.0231$.

Given the deterministic nature of the distance-based QoS model, vehicles typically remain within the optimal range of RSUs. Consequently, although theoretically sound, the distance-based QoS model is rarely invoked in practice within the simulated environment. This design ensures our framework's robustness under varied conditions rather than representing a frequently encountered real-world scenario. Thus, while the model is rigorously designed, its practical application is limited due to strategic management of vehicle positions, preventing significant QoS degradation based on distance.

Load-based QoS

QoS based on RSU load is 100% if the load is within the RSU's capacity, as described in Section 4.7.1. The QoS provided by the RSU directly affects the QoS of all connected vehicles. If the load exceeds capacity, QoS deteriorates proportionally, calculated as:

$$\text{QoS}_{\text{rsu}_j,\text{load}} = \begin{cases} 1 & \text{if load}_j \leq \text{capacity}_j \\ \frac{\text{capacity}_j}{\text{load}_j} & \text{if load}_j > \text{capacity}_j \end{cases}$$

This model is particularly relevant in scenarios where strategies fail to maintain high QoS, especially under limited RSU capacity. Dedicated scenarios within the following sections primarily focus on how the system behaves under these conditions, providing insights into the robustness of various approaches.

Total QoS

The total QoS experienced by a vehicle i connected to RSU j can be obtained by multiplying the distance-based QoS and the load-based QoS:

$$\text{Total QoS}_{v_i} = \text{QoS}_{v_i,\text{dist}} \times \text{QoS}_{\text{rsu}_j,\text{load}}$$

By integrating these validated parameters and models, this approach provides a realistic and practical method for modeling QoS degradation, ensuring robust performance of vehicular networks under varying conditions and high mobility scenarios.

6.1.3 Créteil Roundabout Scenario

The Créteil Roundabout Dataset, presented by Lèbre et al. in [22], provides detailed traffic data for the Europarc roundabout in Créteil, France. This dataset is fundamental

for evaluating local decision-making in ITS and supports hybrid decentralized ITS by providing detailed micro-mobility traces. The dataset can be accessed online. [4].

Road Topology

The Europarc roundabout is a complex urban intersection with the following key features:

- **Six Entrances and Exits**: The roundabout includes six entry and exit points, facilitating high traffic volume from multiple directions.
- **Lane Configuration**: Roads consist of 2-3 lanes, one dedicated bus lane, and four lane-changing spots.

For the experiments, the dataset was reduced to an area of 200x200 meters. This area is sufficient to contain the roundabout and significant portions of the entry and exit streets, allowing the capture of essential vehicle movements within the roundabout. Extending beyond this area would primarily involve straight roads, which are less attractive for examining the strategies' performances and would complicate the analysis.

Figure 6.1 illustrates the road topology of the Europarc roundabout in Créteil, France, which is the focal point for all subsequent experiments using this mobility trace. The visualization in Figure 6.1a is derived directly from the dataset, showcasing all positions where a vehicle is located at any point during the trace analysis. This visualization also indicates the number of lanes in each entry and exit street and within the roundabout. Figure 6.1b provides a real-world reference through an aerial view of the area, sourced from Google Maps, which validates the accuracy of the dataset's road topology representation.

Mobility traces

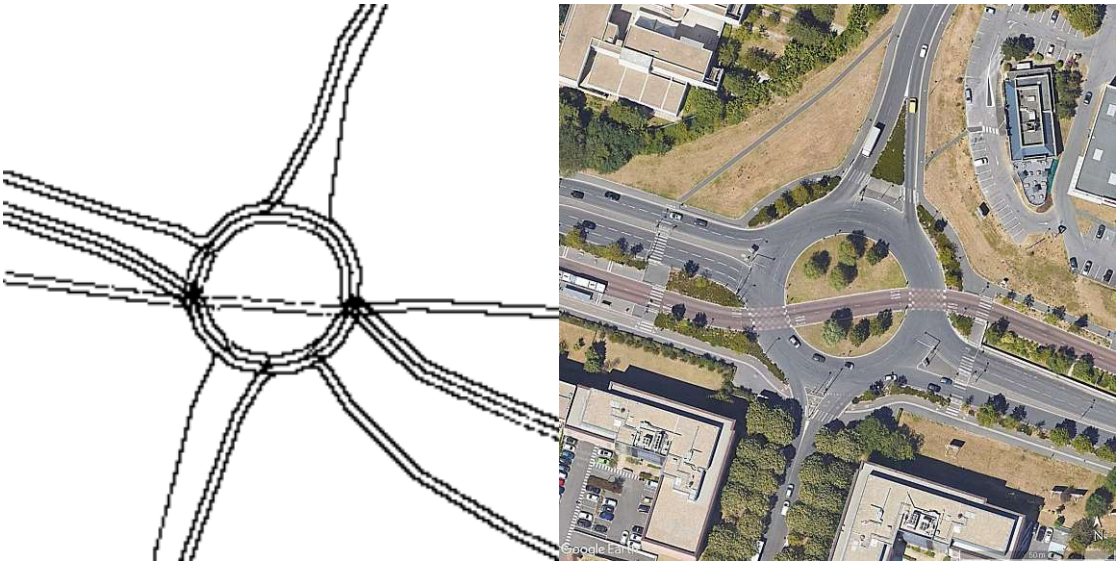
This section compares the mobility traces from the morning and evening rush hours at the Créteil roundabout, providing insights into traffic dynamics during these critical periods.

- **Morning Dataset**

  The morning dataset captures traffic data from 7 AM to 9 AM, covering peak morning rush hours. This period includes approximately 10,000 trips, providing a comprehensive view of traffic dynamics during a critical time frame.

  The number of vehicles in the roundabout over time is analyzed to understand the flow and congestion patterns. Figure 6.2a illustrates the number of vehicles present at different times during the morning dataset period, revealing significant variability in vehicle numbers, with peaks and troughs indicating varying congestion levels.

[4]The Créteil Roundabout Dataset is publicly available at `https://vehicular-mobility-trace.github.io/`, last accessed on August 3, 2024.



(a) Road topology based on vehicle positions (b) Aerial view (Map data ©2024 Google)

Figure 6.1: Europarc Roundabout in Créteil, France

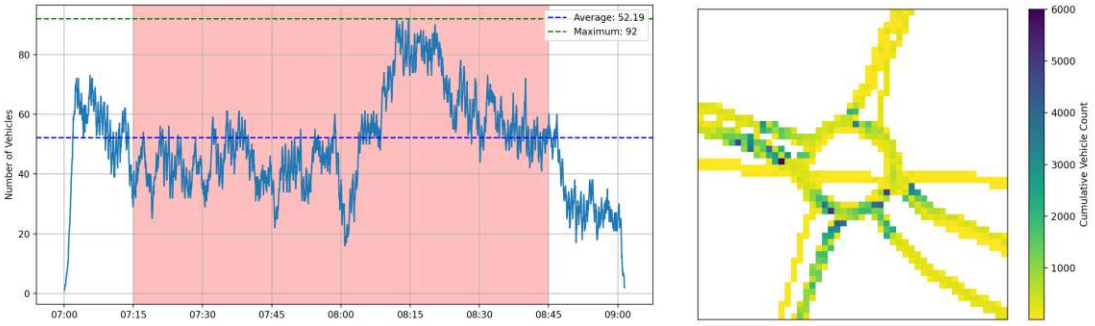
On average, there were 52 vehicles in the roundabout during the experiment period, with a maximum of 92 vehicles observed at the peak time. Over the entire experiment time frame, 3929 distinct vehicle traces were recorded within the space and time constraints of the roundabout. The early and late parts of the dataset (before 7:15 AM and after 8:45 AM) show less representative traffic patterns and are therefore excluded from detailed evaluation. The highlighted red section in Figure 6.2a marks the time window from 7:15 AM to 8:45 AM, selected for further experiments because it reflects stable traffic patterns.

To identify congestion and vehicle accumulation within the roundabout, we generated a map, shown in Figure 6.2b. The map divides the area into 4x4 meter cells, with each cell showing the cumulative vehicle count over all timesteps. This approach highlights traffic hotspots, particularly at the entry roads and just before the roundabout, where most traffic lights are located. The color intensity reflects the traffic density, offering clear insights into areas most prone to congestion during the morning rush hour.

- **Evening Dataset**

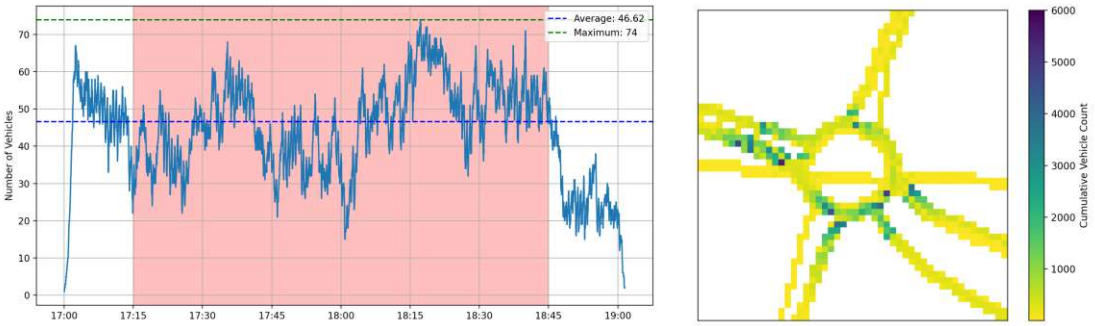
  The evening dataset captures traffic from 5 PM to 7 PM, covering peak evening rush hours. Like the morning dataset, this period includes many trips, providing insights into the traffic dynamics during another critical time frame.

  Figure 6.3a illustrates the number of vehicles present at different times during the evening dataset. The data shows fluctuations in vehicle numbers, with notable peaks indicating periods of high congestion.



(a) Number of Vehicles Over Time (b) Traffic Density Map

Figure 6.2: Créteil Roundabout, Morning Dataset



(a) Number of Vehicles Over Time (b) Traffic Density Map

Figure 6.3: Créteil Roundabout, Evening Dataset

On average, there are approximately 47 vehicles in the roundabout during the evening experiment period, with a maximum of 74 vehicles observed at the peak time. Throughout the entire evening time frame, 3881 distinct vehicle traces were recorded within the space and time constraints of the roundabout. The highlighted red section in Figure 6.3a marks the time window from 5:15 PM to 6:45 PM, chosen for experiments due to its consistent traffic flow.

To identify areas of congestion during the evening rush hour, we generated a map similar to the one used for the morning dataset, as shown in Figure 6.3b.

Comparing the morning and evening datasets reveals distinct traffic patterns at the Créteil roundabout during peak hours. The morning dataset, with an average of 52 vehicles and a maximum of 92 vehicles, indicates a higher level of congestion than the evening dataset, which has an average of 47 vehicles and a maximum of 74 vehicles. Moreover, the total vehicle counts show a slight difference, with 3929 distinct vehicle traces recorded in the morning and 3881 in the evening. This slight difference in the total number of distinct vehicle traces and the more pronounced difference in average and

maximum concurrent vehicles likely reflects increased traffic congestion in the roundabout area during the morning rush hour.

Both datasets show significant variability in vehicle numbers, but the morning rush hour experiences more pronounced peaks and a higher overall traffic load. This disparity may be attributed to various factors, such as the greater urgency for individuals to reach their destinations in the morning or differing traffic management strategies during these times. As described in the paper by Lèbre et al. [22], morning traffic tends to show higher peaks due to the synchronized rush of commuters heading to work. In contrast, evening traffic is often more dispersed as people leave work at varying times. The study highlights that morning congestion is worsened by the collective movement of individuals toward their workplaces, leading to more significant bottlenecks and higher vehicle counts at any given time. This higher number of concurrent vehicles is also noticeable in the density maps, where more pronounced congestion is evident at the entry roads and traffic lights during the morning period. Conversely, evening traffic disperses longer as commuters leave work at different times, resulting in lower average and peak vehicle numbers, and slightly less severe congestion at these critical points.

Utilizing the morning and evening datasets provides a comprehensive understanding of daily traffic dynamics and enables comparing different traffic conditions. This approach ensures the development of tailored and effective traffic management solutions validated by real-world data and creates two advantageous 90-minute mobility traces for all further experiments.

Sparse RSU configuration

The sparse RSU configuration is designed with four RSUs, each with a range of 70 meters. These RSUs are strategically positioned to cover the roundabout's primary entry and exit points, ensuring that vehicles experience multiple handovers during their traversal. No RSU is placed directly in the roundabout's center. Instead, the RSUs are aligned along the middle of the four main entry/exit streets, ensuring comprehensive coverage of the road segments and maintaining 100% QoS. This alignment guarantees full coverage of the critical road segments, thereby maintaining a hypothetical 100% distance-based QoS. The layout is depicted in Figure 6.4.

This configuration is designed to assess the basic functionality and efficiency of RSU-based handover coordination within a VEC environment. Given the setup, vehicles will experience multiple handovers as they traverse the roundabout. This scenario tests the system's ability to manage handovers efficiently and maintain QoS. In addition, we can measure the impact that an increased load-sharing interval has on handover coordination efficiency.

As analyzed above, the maximum number of vehicles within the roundabout at any given time is 92. Each vehicle generates approximately 1.54 TFLOPS of computational load, as detailed in Section 6.1.1. This results in a total computational demand of approximately 141.68 TFLOPS, which is roughly equivalent to the capacity of 2.2 NVIDIA Tesla T4

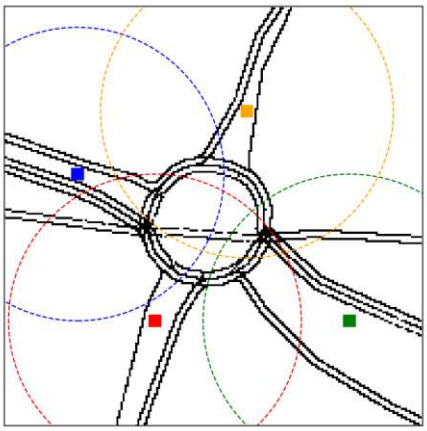


Figure 6.4: Créteil Roundabout Sparse Configuration with 4 RSUs

GPUs (each offering 65 TFLOPS). Equipping all four RSUs with Tesla T4 GPUs ensures that each vehicle can potentially receive 100% load-based QoS under optimal conditions.

Experiments will also be conducted with halved RSU capacities to assess the system's robustness and the effectiveness of the handover coordination under constrained conditions. Each RSU operates with 32.5 TFLOPS of capacity in this scenario. This reduced capacity will likely result in situations where not all vehicles can be provided with 100% QoS, particularly during high traffic density or localized congestion. These experiments will offer insights into the system's performance under limited RSU resources.

Dense RSU configuration

The dense RSU configuration enhances the sparse setup by increasing the number of RSUs to nine, each with a range of 70 meters. This configuration includes the original four RSUs, with an additional four positioned between them, creating a circular coverage pattern. A ninth RSU is placed in the roundabout's center. This arrangement ensures that each road segment is covered by at least two RSUs, with most segments receiving coverage from three or more RSUs. The layout is illustrated in Figure 6.5.

This dense configuration evaluates system performance under a more robust RSU deployment, focusing on enhanced coverage, reduced handover frequency, and improved QoS. By increasing the number of RSUs, we expect to see improvements in load distribution and a more significant reduction of the number of handovers compared to the baseline approaches, compared with the 4 RSU setup.

Equipped with NVIDIA Tesla T4 GPUs providing 65 TFLOPS each, the dense configuration's combined capacity of 575 TFLOPS exceeds the computational demand of 141.68 TFLOPS generated by a maximum of 92 vehicles. This surplus capacity is expected to maintain 100% load-based QoS even under peak conditions.

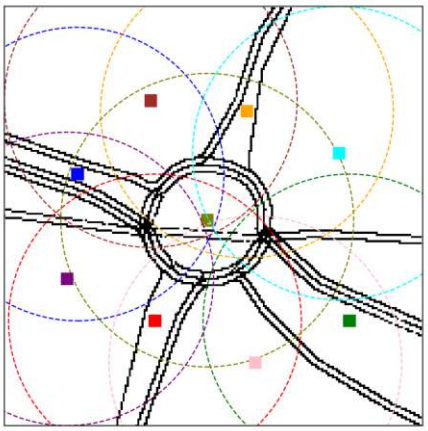


Figure 6.5: Créteil Roundabout Dense Configuration with 9 RSUs

To further assess the system's robustness, experiments will be conducted under two constrained scenarios:

- **Halved Capacity**: Each RSU operates at 32.5 TFLOPS, providing a total system capacity of 292.5 TFLOPS.
- **Quartered Capacity**: Each RSU operates at 16.25 TFLOPS, providing a total system capacity of 146.25 TFLOPS.

These experiments will help determine how the system performs under varying load conditions and identify the capacity thresholds at which QoS begins to degrade.

6.1.4 RS Agent Strategy Parametrization

This section focuses on determining the optimal strategy constants for the RS agents through a series of experiments. All experiments were performed with the highest possible load-sharing frequency, set to 1, meaning load information is shared after each timestep.

Scenario for Parameter Tuning

The experiments utilized the Créteil morning scenario with a 4 RSU setup, each operating at halved capacity, as described in Section 6.1.3. This configuration was selected due to its challenging environment, characterized by high traffic density and limited RSU capacity, making it an effective stress test for the strategies. By pushing the system toward its operational limits, we can identify parameters that ensure robust performance even under less favorable conditions.

This sparse setup forces the RS agents to make critical handover and load-balancing decisions, essential for identifying optimal strategy parameters. It represents the most

demanding scenario the RS agents might encounter, ensuring that the selected parameters will be effective across various configurations.

Once optimized under these challenging conditions, these parameters were applied to all subsequent experiments, including those with the denser configuration of nine RSUs. This approach ensures that the parameters are effective in high-stress scenarios and adaptable to varying network densities, guaranteeing consistent QoS and efficient load balancing in all configurations.

The following parameters for the proposed strategy, detailed in Section 5.2, were tuned:

- **Leaving Threshold**: Percentage of the RSU's range at which the RSU will hand over the vehicle if it is outside the inner part of the range. For example, if the threshold is 90% and the range is 50 meters, the vehicle will be handed over if it is more than 45 meters away from the RSU.
- **Overload Threshold**: Percentage of the RSU's capacity at which handovers of the most suitable vehicles will be forced when the load percentage exceeds this threshold.
- **Load-Balancing Hysteresis**: Margin for utilization percentage difference that another RSU must be lower than the current one to justify a handover. This margin is applied to either an alternative handover to a less utilized RSU or an imperative handover if the current RSU is overloaded. The load of the considered vehicle is added to the other RSU's load before comparison.
- **Load-Balancing Minimum Suitability**: Minimum suitability another station should have for a vehicle to be considered for a handover, based on the suitability score presented in Section 5.2.1. This is ignored if the source RSU is overloaded.

General Observations

1. **Leaving Threshold**: Reducing the leaving threshold improves all metrics, with setting the leaving range to 0 yields the best results across the board, replicating the latest Handover (HO) strategy for mobility-caused handovers.
2. **Overload-triggered Handover**: Overload-triggered handovers are infrequently used, as load-balancing effectively addresses these situations in nearly all evaluated strategy configurations.
3. **Handover Frequency**: Minimizing the number of handovers requires a combination of high load-balancing hysteresis, high minimum suitability, and a high overload threshold.
4. **Maximizing QoS**: Both average and minimum QoS are maximized with minimal load-balancing hysteresis.

5. **Minimizing Gini Coefficient**: Achieving a low Gini coefficient requires minimal load-balancing hysteresis, overload threshold, and suitability.

6. **Visualization**: Visual analysis of specific configurations revealed the causes of unexpected behaviors and aided in identifying implementation bugs.

Finding the Best Parameters

Identifying the best parameters in this search space was challenging. Two evaluation metrics were introduced, relevant only to the parameter selection:

- **Normalization Process**
  - Each criterion's score is normalized to a value between 0 and 1.
  - For handover count and Gini coefficient, where lower values are better, the normalization formula used is:

$$\text{Normalized Score} = \frac{\text{Minimum Observed Value}}{\text{Observed Value}}$$

  - For average QoS and minimum QoS, where higher values are better, the normalization formula used is:

$$\text{Normalized Score} = \frac{\text{Observed Value}}{\text{Maximum Observed Value}}$$

  - This normalization ensures that the best possible score for each criterion is 1, and lower scores reflect worse performance relative to the best-observed performance.

- **Evaluation Scores**
  - **Evaluation Sum**: The sum of normalized scores for handover count, average QoS, minimum QoS, and Gini coefficient.
  - **Evaluation Product**: The product of the same normalized scores, which is more sensitive to outliers.

The normalized scores range between 0 and 1, with 1 representing the best possible score. The top parameter configurations, as determined by both metrics, are shown in Table 6.3.

These results indicate that a load-balancing hysteresis of 0.05 is the most effective strategy. The optimal overload threshold is 0.7, while a load-balancing minimum suitability of 0.3 emerges as the top choice, closely aligning with the average of the highest-performing values. This combination provides a balanced and efficient approach for managing handovers and maintaining QoS in the tested scenarios.

Table 6.3: Top RS Agent Strategy Parameter Configurations

Overload	Hysteresis	Suitability	Eval Sum	Eval Product
0.7	0.05	0.3	3.8175	0.8252
0.6	0.05	0.35	3.8159	0.8236
0.8	0.05	0.25	3.8151	0.8232
0.9	0.05	0.2	3.8137	0.8216
0.8	0.05	0.2	3.8137	0.8215

Chosen Parameters for RS Agent Strategy

Based on the analysis, the following RS agent strategy parameters were selected for all subsequent experiments:

- **Leaving Threshold**: 0
- **Overload Threshold**: 0.7
- **Load-Balancing Hysteresis**: 0.05
- **Load-Balancing Minimum Suitability**: 0.3

These parameters will be used to evaluate the performance of the RS agent strategies in subsequent experiments, ensuring a balance between minimizing handovers, maximizing QoS, and maintaining fairness in load distribution.

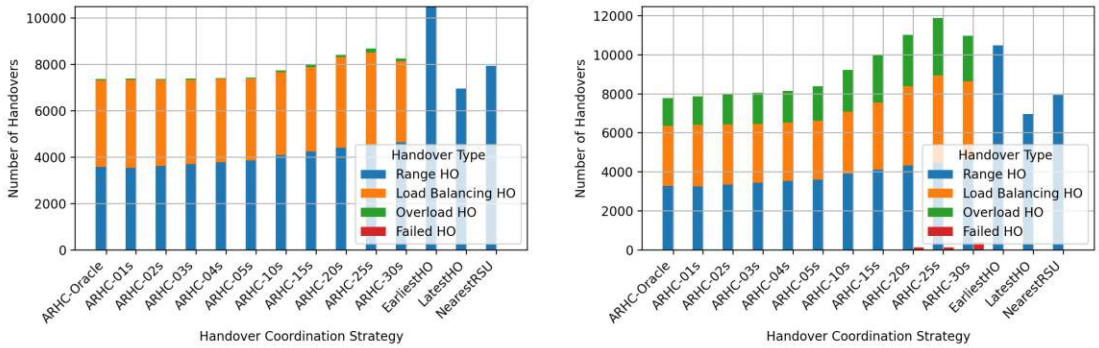
6.2 Results

6.2.1 Experiment 1: Créteil Roundabout - Sparse RSU Configuration

In this section, we assess the performance of a sparse RSU configuration at the Créteil roundabout during morning and evening traffic. This analysis focuses on the isolated performance of the sparse setup, examining handover frequency, QoS metrics, and load balancing under varying conditions. We analyze the system using four RSUs under full-capacity and half-capacity conditions to understand the impact of reduced RSU capacity on handover coordination within our proposed ARHC strategy and in comparison to baseline strategies.

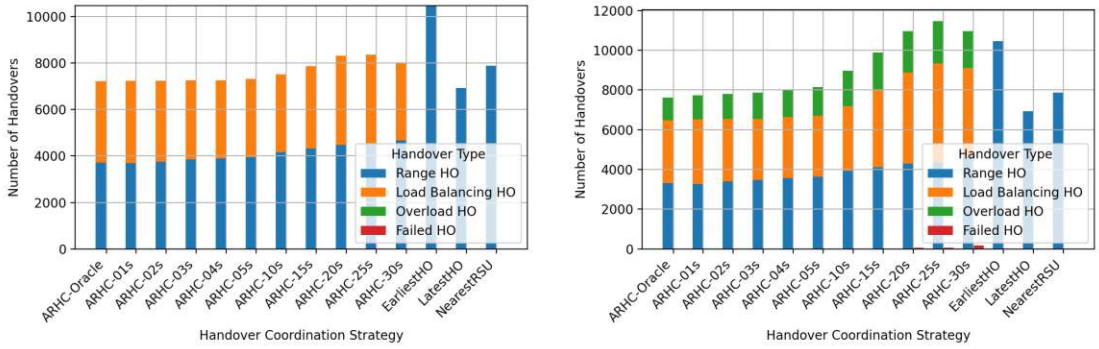
Additionally, we examine the impact of decreasing the frequency of load sharing for both ARHC and baseline strategies. ARHC is further evaluated using a load-sharing oracle that provides real-time load information. For clarity, ARHC strategies are labeled with the respective load-sharing interval in seconds (e.g., ARHC-05s for a 5-second interval).

It is important to note that the morning mobility trace exhibits a higher average and maximum number of concurrent vehicles than the evening trace but similar numbers



(a) Full RSU Capacity (b) Half RSU Capacity

Figure 6.6: Number of Handovers of Créteil Sparse Configuration at Morning



(a) Full RSU Capacity (b) Half RSU Capacity

Figure 6.7: Number of Handovers of Créteil Sparse Configuration at Evening

of total vehicles, as described in Section 6.1.3. The reduction in vehicle numbers from morning to evening indirectly simulates an increase in RSU capacity, offering a comprehensive evaluation of the strategies' performance and adaptability under varying traffic densities and RSU capacities.

Handover Frequency

Figures 6.6 and 6.7 present the number of handovers for full and half RSU capacity configurations during the morning and evening mobility traces. Baseline strategies (earliest/latest HO and nearest RSU) are compared alongside ARHC, with load-sharing intervals increasing from left to right.

Handovers are classified as successful or failed, with successful handovers further categorized by their triggers: range-based, load-balancing, and overload-triggered.

In the morning scenario at full RSU capacity (Figure 6.6a), the latest HO strategy results in the fewest handovers, while the nearest RSU strategy performs slightly worse than

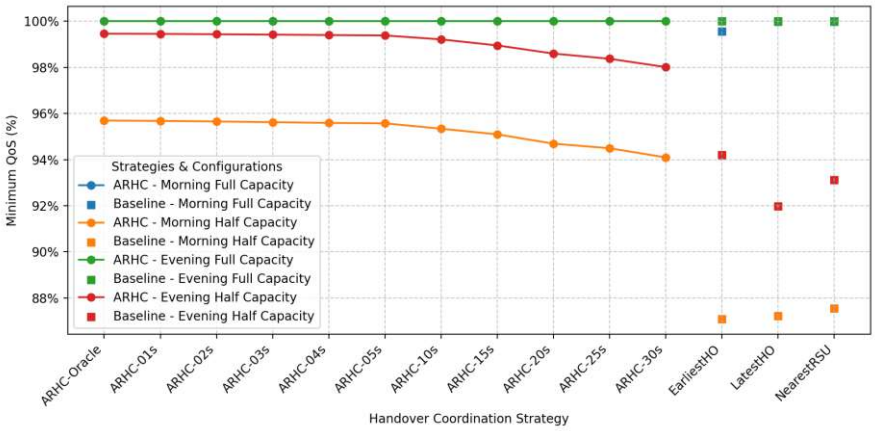


Figure 6.8: Minimum QoS Scores of Créteil Sparse Configurations

ARHC at low load-sharing intervals. The earliest HO strategy generates the highest number of handovers. As load-sharing intervals increase, the total number of handovers rises, driven primarily by more range-based handovers. Importantly, no failed handovers and almost no overload-triggered occur, indicating that the high RSU capacity effectively prevents overloads and ensures reliable handovers.

Baseline strategies maintain consistent handovers regardless of capacity when RSU capacity is reduced to half (Figure 6.6b). In contrast, ARHC sees an increase in handovers compared to the full-capacity scenario, partly related to overload-triggered handovers. The latest HO strategy continues to result in fewer handovers than ARHC, while the earliest HO strategy sometimes surpasses certain ARHC configurations. As load-sharing intervals grow, the total number of handovers increases across all types, with a few failed handovers occurring, although these remain significantly lower than successful ones.

The evening scenario (Figures 6.7) shows a similar pattern to the morning, with generally fewer handovers due to lower traffic volumes. The relative performance of the strategies and the impact of load-sharing intervals are consistent with the morning findings.

QoS Metrics

In this section, we analyze minimum and average QoS values under different configurations and strategies, comparing the performance of ARHC with baseline strategies across various load-sharing intervals. The goal is to assess how well each strategy maintains QoS in high-capacity and reduced-capacity scenarios during morning and evening traffic. The results for minimum QoS are presented in Figure 6.8, while the average QoS results are shown in Figure 6.9.

At full RSU capacity, ARHC consistently achieves perfect minimum QoS across all load-sharing intervals for both mobility traces. The baseline strategies also perform similarly, except for the earliest HO strategy in the morning trace. These excellent scores

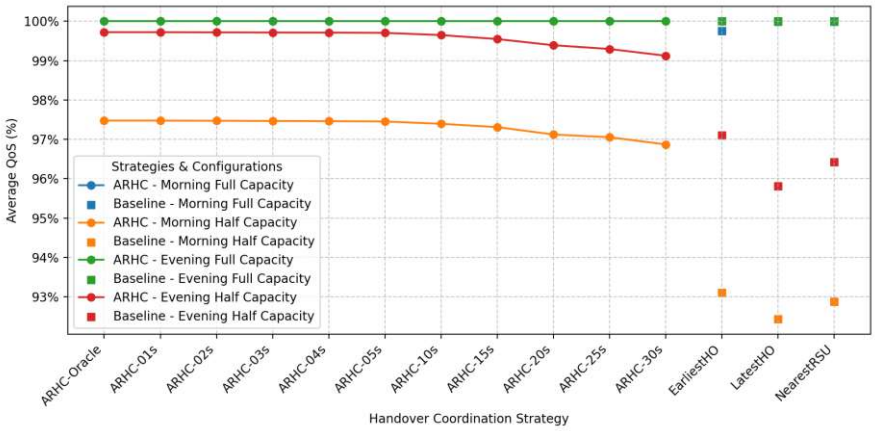


Figure 6.9: Average QoS Scores of Créteil Sparse Configurations

are attainable because the RSU capacity significantly exceeds the load generated by vehicles, both overall and within local mobility hotspots. The average QoS results mirror this trend, showing slightly higher values due to the averaging process, thus underscoring the network's stability in these scenarios.

When the RSU capacity is reduced to half, the results show a general decline in minimum and average QoS. In the evening scenario, ARHC performs slightly worse than full-capacity scenarios, particularly at low load-sharing intervals. However, with its higher concurrent load, the morning trace shows a significant drop in performance at half RSU capacity. The baseline strategies exhibit even greater reductions in QoS, with ARHC maintaining superior performance across both mobility traces, even at the highest load-sharing intervals considered.

Load Balancing Efficiency

This section evaluates load balancing across different strategies and configurations using the Gini coefficient, where 0 indicates perfect balance, and 1 represents maximum imbalance. By analyzing these values, we can assess how well each strategy maintains an even distribution of computational load among RSUs. Notably, the Gini values for baseline strategies remain consistent across all RSU capacity configurations, as they do not adapt their load balancing based on capacity.

All Gini results are consolidated in Figure 6.10.

The analysis reveals several key insights. First, ARHC performs strongly at low load-sharing intervals, achieving Gini values between 0.1 and 0.15, reflecting effective load balancing. However, as load-sharing intervals increase, the balance deteriorates, resulting in higher Gini values.

Interestingly, load balancing worsens with increased RSU capacity and higher traffic volumes. For instance, the morning scenario, which has more concurrent vehicles, exhibits

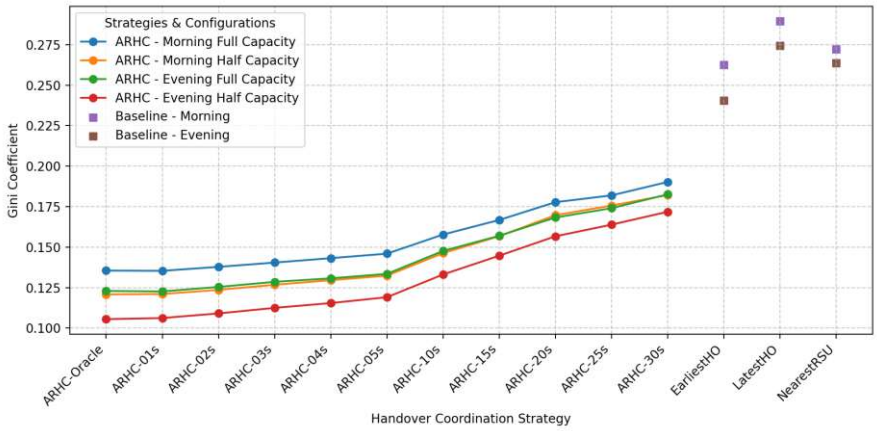


Figure 6.10: Gini Scores of Créteil Sparse Configurations

poorer load balancing than the evening scenario, and half-capacity RSU configurations outperform full-capacity setups in maintaining balanced loads.

Baseline strategies consistently underperform compared to ARHC in load balancing, even at the highest load-sharing intervals. Among these, the earliest HO strategy fares the best, though it still lags significantly behind ARHC.

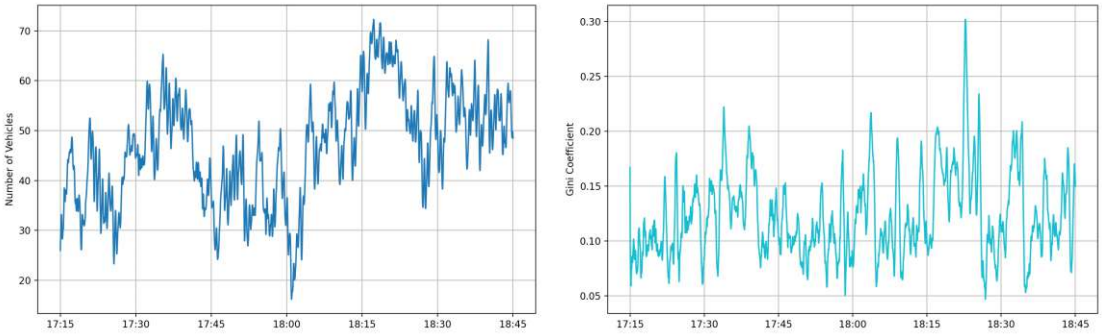
Time-Based Analysis

After presenting the broader evaluation of RSU capacity configurations and load-sharing intervals for the Créteil Sparse configuration, we now focus on the performance of the proposed ARHC strategy in two specific scenarios: the best-case and worst-case situations in terms of RSU capacity and vehicular load. These scenarios are evening traffic at full RSU capacity (best-case) and morning traffic at half RSU capacity (worst-case). Both scenarios use the ARHC strategy with an optimal load-sharing interval of 1 second for consistency.

- **Best-Case Scenario: Evening Traffic at Full RSU Capacity**

  The evening traffic scenario exhibits significant variability, with major traffic peaks around 5:40 PM, 6:20 PM, and 8:00 PM, as shown in Figure 6.11a. The 6:20 PM peak leads to a noticeable increase in load imbalance, with the Gini coefficient temporarily spiking to 0.3 (Figure 6.11b). Despite these peaks, the ARHC strategy effectively manages load distribution, keeping the Gini coefficient primarily between 0.05 and 0.15. These imbalances are mainly due to congestion in areas serviced by a single RSU, which challenges the system's ability to distribute the load evenly.

  During periods of lower traffic, the ARHC-01s strategy successfully maintains balanced load distribution, demonstrating its adaptability to fluctuating traffic conditions. The congestion observed during peak times (Figure 6.3b) aligns with



(a) Vehicles Over Time (10s Rolling Mean) (b) Gini Scores Over Time (30s Rolling Mean)

Figure 6.11: ARHC-01s Performance Over Time (Créteil, Evening, Sparse, Full Capacity)

specific areas within the RSU setup (Figure 6.4), highlighting the challenge of achieving load balance in high-traffic zones. Despite the traffic peaks, the RSU capacity is handled efficiently, resulting in consistently high QoS throughout the scenario, which negates the need for further QoS analysis.

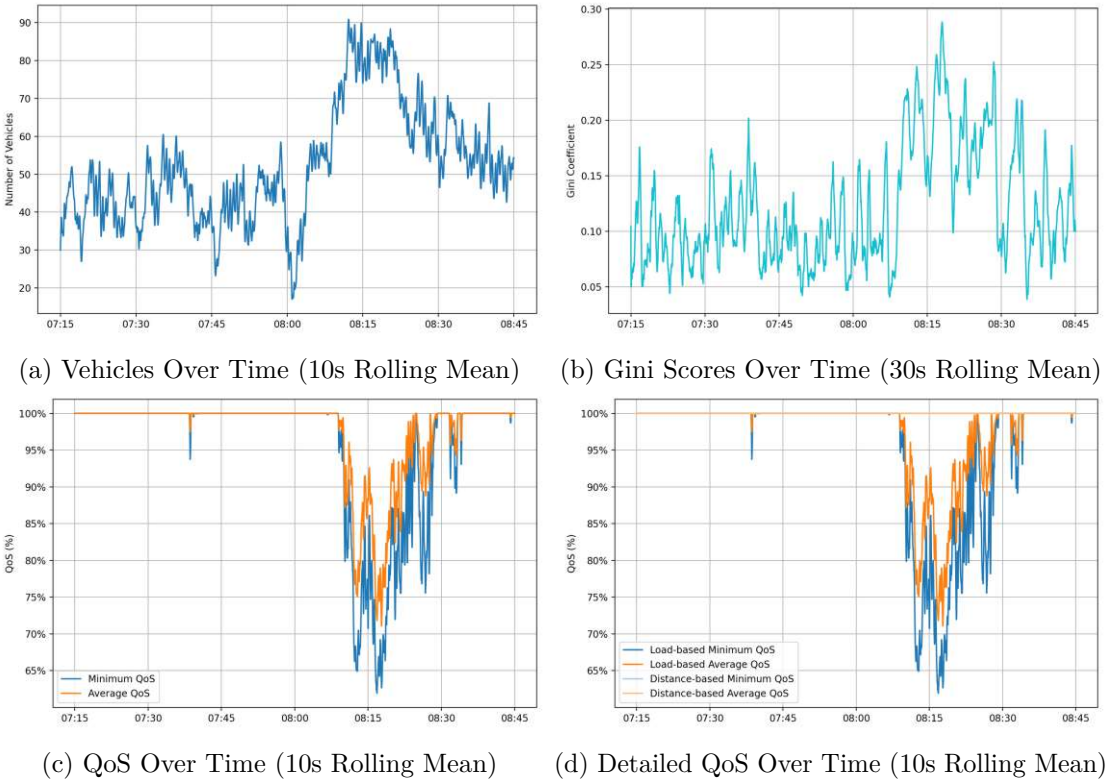
- **Worst-Case Scenario: Morning Traffic at Half RSU Capacity**

  In the morning traffic scenario with half RSU capacity, vehicle counts fluctuate significantly, with a noticeable drop around 8:00 AM followed by a sharp increase, peaking at over 90 concurrent vehicles within 15 minutes (Figure 6.12a). This rapid traffic increase is accompanied by a corresponding rise in load imbalance, with the Gini coefficient approaching 0.3 between 8:10 and 8:35 AM (Figure 6.12b). Outside these peak times, the Gini values generally remain within a range of 0.05 to 0.15, with occasional spikes reaching around 0.2.

  Due to the higher traffic load and reduced RSU capacity, the QoS experiences significant stress, as shown in Figure 6.12c. As traffic exceeds approximately 60 vehicles, QoS begins to degrade noticeably, with minimum values dropping below 65% during the peak when traffic surpasses 90 vehicles. Detailed analysis confirms that the overall QoS drop is primarily due to RSU overload, as distance-based QoS remains stable at 1 (Figure 6.12d). This stability indicates that the ARHC strategy effectively maintains distance-based QoS even under high load conditions.

  In summary, the QoS decreases proportionally with traffic increases beyond the 60-vehicle threshold, clearly reflecting the system's response to peak traffic under reduced capacity.

In both scenarios, the ARHC strategy effectively manages load distribution, with the Gini coefficient remaining stable overall. In the best-case scenario, Gini spikes are brief and occur only during short periods of high traffic. In contrast, the worst-case scenario shows more pronounced fluctuations in the Gini coefficient during extended periods of



(a) Vehicles Over Time (10s Rolling Mean)

(b) Gini Scores Over Time (30s Rolling Mean)

(c) QoS Over Time (10s Rolling Mean)

(d) Detailed QoS Over Time (10s Rolling Mean)

Figure 6.12: ARHC-01s Performance Over Time (Créteil, Morning, Sparse, Half Capacity)

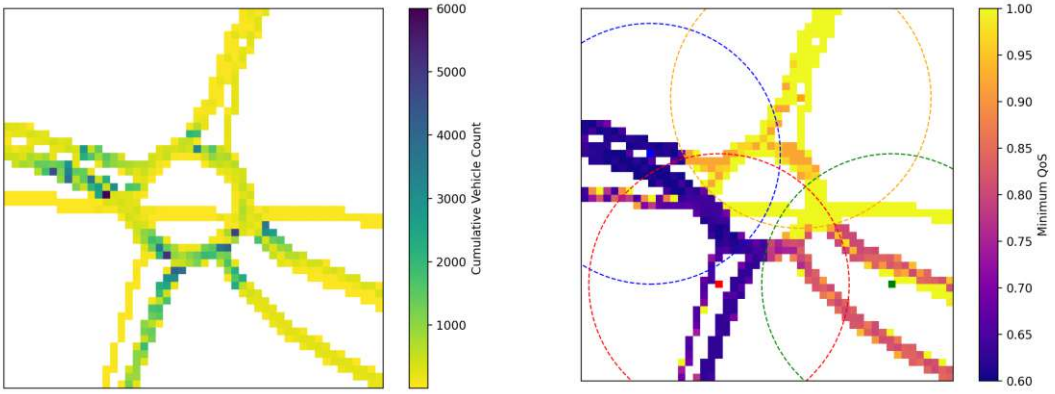
extremely high traffic, highlighting the increased challenge of maintaining load balance under sustained pressure.

The key difference emerges in QoS performance. In the best-case scenario, QoS remains consistently high, even with occasional Gini peaks. However, in the worst-case scenario, QoS drops significantly when traffic exceeds the 60-vehicle threshold for an extended period, falling below 65% due to RSU overload.

Traffic Density and QoS

This section examines how traffic density affects QoS in the Créteil Roundabout scenario, focusing on morning traffic with half RSU capacity and a load-sharing interval of 1 second. Understanding these effects helps identify network performance challenges and areas for potential improvement, particularly in the context of the ARHC strategy's ability to manage handovers and optimize QoS under these conditions.

The previously introduced Traffic Density Map (Section 6.1.3), visualized in Figure 6.13a, shows the distribution of vehicles during the morning rush hour. It highlights areas of high congestion, particularly near entry roads and before the roundabout, where delays



(a) Traffic Density Map

(b) Minimum QoS Map

Figure 6.13: Créteil Roundabout: Traffic Density and QoS Maps (Sparse Configuration, Half Capacity, Morning)

occur due to traffic lights, with congestion especially noticeable at the western and southern entrances and to a lesser extent at the eastern entry.

To evaluate the impact on minimum QoS, a corresponding Minimum QoS Map is used, which is visualized in Figure 6.13b. This map illustrates the variation in QoS across the roundabout, with darker areas indicating significant drops in QoS that often align with the high-density regions identified in the Traffic Density Map.

A clear pattern emerges: Areas with the highest traffic density often align with regions where QoS significantly drops. This correlation suggests that heavy congestion directly impacts the network's ability to maintain service quality, particularly where RSUs are under the most strain.

The most significant QoS reductions occur at the southern and western exits/entries and the south-western roundabout area, primarily covered by the blue and red RSUs. Here, QoS drops to around 0.6, indicating that these RSUs were operating at over 166% of their capacity during peak times. This widespread drop in minimum QoS highlights the areas of the network most affected by congestion. It underscores the importance of the ARHC strategy's role in mitigating these effects, even if it struggles to maintain optimal performance under such high loads.

In contrast, the eastern area covered by the green RSU and partially overlapping with the red RSU shows a less severe QoS drop, while the northern area served by the yellow RSU maintains near-perfect QoS throughout the simulation. This suggests that RSUs in less congested areas are better able to handle the load, with the ARHC strategy successfully managing handovers and load distribution to maintain service quality even with reduced capacity.

Overall, this analysis shows that high traffic density is a key factor in QoS degradation, particularly under reduced RSU capacity. These findings emphasize the importance of

strategic RSU placement and capacity planning to mitigate the effects of congestion on service quality, which is critical for optimizing the ARHC strategy and improving network performance in high-traffic scenarios.

These findings highlight the sparse configuration's strengths and limitations, particularly its ability to manage QoS under varying capacities. The cross-experiment analysis Section 6.3 will present a comparative analysis of these results with those from the dense configuration.

6.2.2 Experiment 2: Créteil Roundabout, Dense Configuration

In this section, we examine the performance of the dense RSU configuration at the Créteil roundabout, focusing specifically on its capacity to handle higher traffic volumes and more complex handover scenarios. The experiment evaluates handover frequency, QoS, and load balancing efficiency under various traffic conditions. The dense configuration, featuring 9 RSUs, is detailed in Section 6.1.3 and illustrated in Figure 6.5. This setup provides a more resource-rich environment than the sparse configuration, allowing us to explore how different capacity levels affect handover performance, QoS, and load balancing under varying vehicular loads.

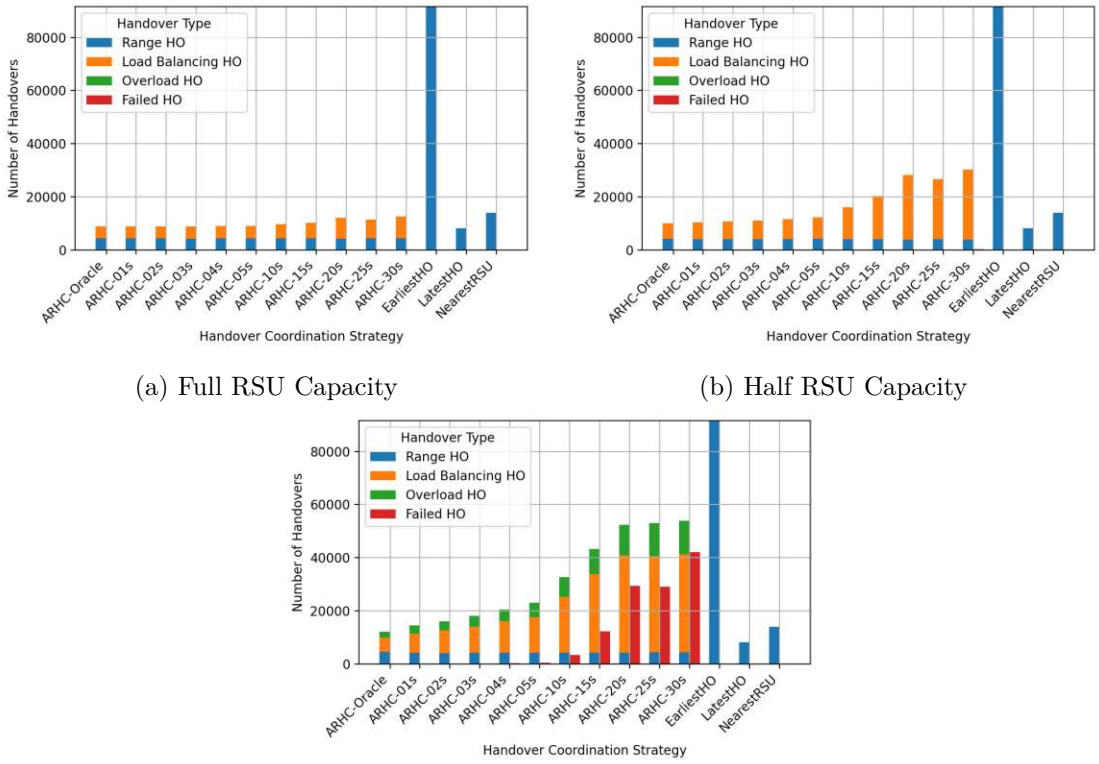
Handover Frequency

Figures 6.14 and 6.15 illustrate the number of handovers across full, half, and quarter RSU capacities for both morning and evening traces under the dense RSU configuration.

A consistent pattern emerges across both morning and evening traces: The earliest HO strategy consistently requires the highest number of handovers, significantly more than any other strategy. This trend is evident across all capacity configurations, though it is less pronounced at lower capacities.

The latest HO strategy consistently minimizes handovers, outperforming the other strategies regardless of the mobility trace. ARHC generally performs better than the nearest RSU strategy at full RSU capacity and remains competitive at low load-sharing intervals under half capacity. However, its performance deteriorates as load-sharing intervals increase, with most handovers attributed to load balancing in these cases. Notably, overload-related and failed handovers only occur in the quarter-capacity configuration, mainly when load-sharing is less frequent. This issue is more pronounced than in the sparse configuration.

Comparing the morning and evening traces, the results are generally similar. However, the evening trace shows slightly fewer overload-related handovers and failed handover attempts, especially in the quarter-capacity scenario. This is primarily due to the reduced load from having fewer concurrent vehicles in the evening, alleviating some of the strain on the RSUs.



(a) Full RSU Capacity

(b) Half RSU Capacity

(c) Quarter RSU Capacity

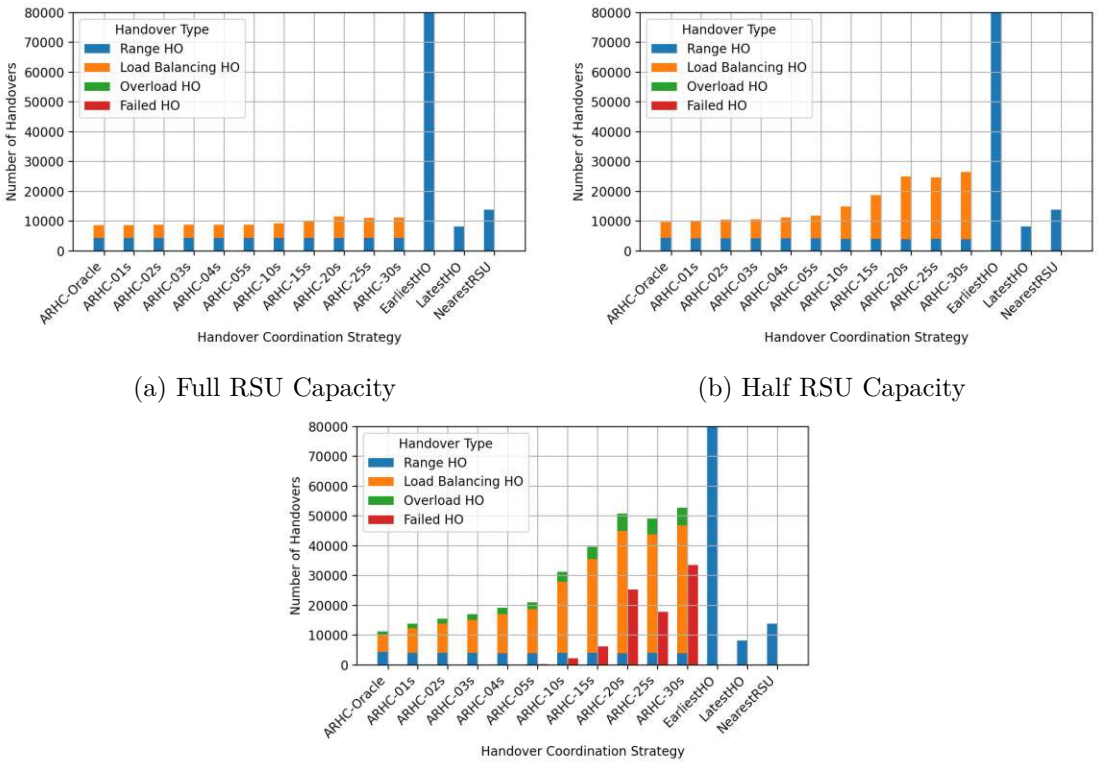
Figure 6.14: Number of Handovers of Créteil Dense Configuration at Morning

QoS Metrics

The dense RSU configuration at the Créteil roundabout reflects trends similar to those of the sparse setup. At full RSU capacity, both ARHC and baseline strategies achieve optimal values. However, as RSU capacity decreases, ARHC outperforms baseline strategies, particularly under high-traffic conditions. Even at quarter capacity, ARHC maintains near-perfect minimum QoS, though its performance slightly dips during more congested morning traces, as shown in Figure 6.16. This dip becomes more pronounced in scenarios with higher load-balancing intervals, particularly under heavier traffic.

The average QoS metric, visualized in Figure 6.17, follows a similar trend but with generally higher scores across all strategies. While ARHC remains near-optimal in most scenarios, it shows a slight decline in the morning trace at quarter capacity, where the highest load-to-RSU-capacity ratio is encountered. Baseline strategies perform optimally at full capacity, with the earliest HO strategy remaining almost optimal at half capacity, whereas other baseline strategies demonstrate weaker performance at lower capacities.

The dense configuration presents unique challenges and benefits, particularly in managing handover frequency and load balancing under reduced capacity. These results will



(a) Full RSU Capacity

(b) Half RSU Capacity

(c) Quarter RSU Capacity

Figure 6.15: Number of Handovers of Créteil Dense Configuration at Evening

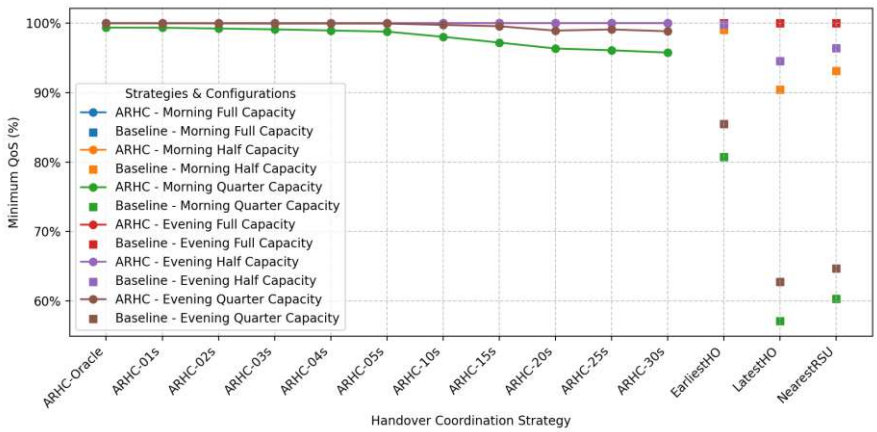


Figure 6.16: Minimum QoS Scores of Créteil Dense Configurations

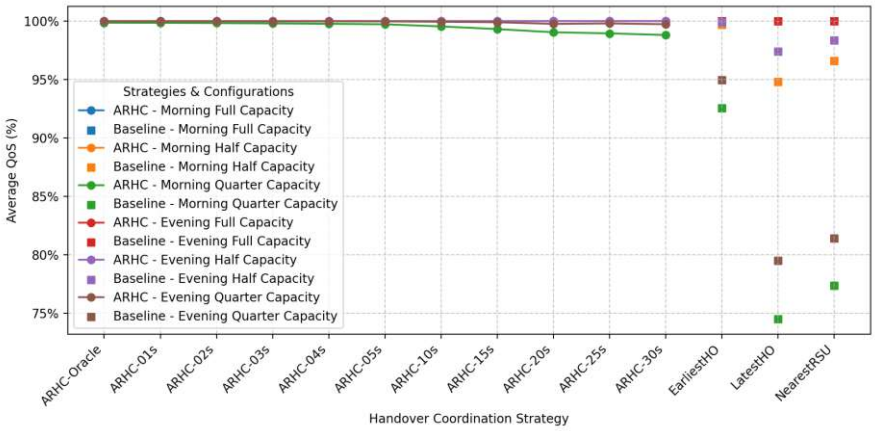


Figure 6.17: Average QoS Scores of Créteil Dense Configurations

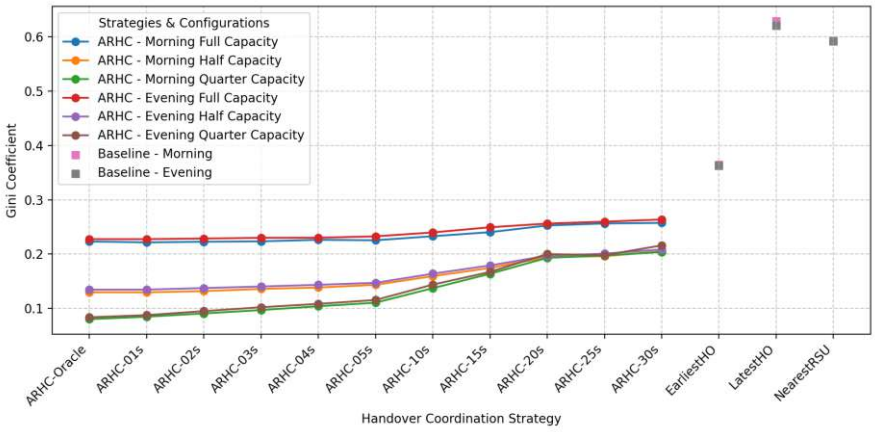
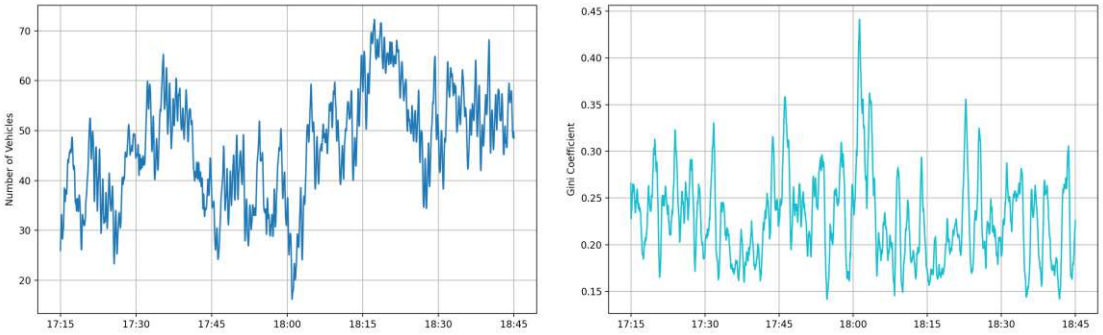


Figure 6.18: Gini Scores of Créteil Dense Configurations

be directly compared with those of the sparse configuration in the subsequent cross-experiment analysis section.

Load Balancing Efficiency

When examining load balancing (in)equality, as shown in Figure 6.18, it is evident that the choice between the morning or evening mobility trace has minimal impact across all strategies. Similar to the sparse configuration, a counterintuitive pattern emerges: ARHC achieves greater load balancing equality in more resource-constrained scenarios. Higher load-sharing intervals consistently lead to increased imbalance, with this effect being more pronounced in lower-capacity configurations. Baseline strategies perform significantly worse than ARHC, even at the highest load-sharing intervals, though the earliest HO strategy remains the best performer.



(a) Vehicles Over Time (10s Rolling Mean) (b) Gini Scores Over Time (30s Rolling Mean)

Figure 6.19: ARHC-01s Performance Over Time (Créteil, Evening, Dense, Full Capacity)

Time-Based Analysis

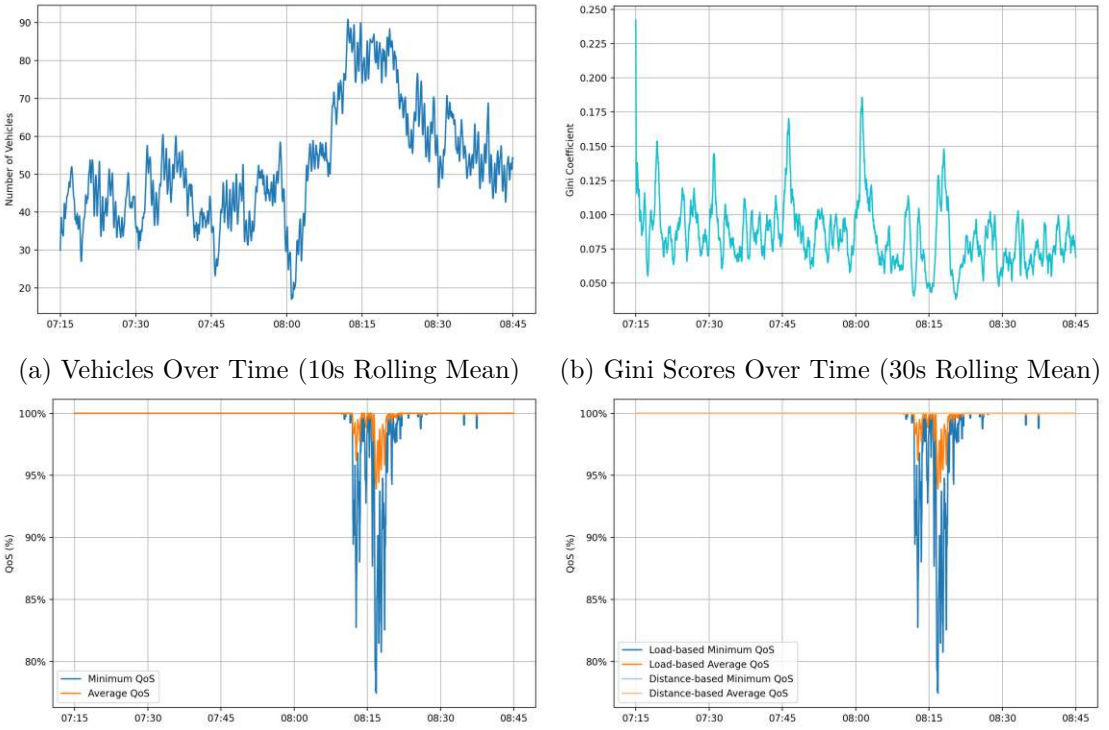
In the dense RSU configuration, the presence of 9 RSUs, with overlapping coverage areas, introduces additional complexity in managing load distribution and maintaining QoS. The time-based analysis provides insight into how the ARHC strategy performs under varying traffic conditions, particularly in scenarios with different RSU capacities.

- **Best-Case Scenario: Evening Traffic at Full RSU Capacity**

  In the best-case scenario, with evening traffic and full RSU capacity, the Gini coefficient remains mostly between 0.15 and 0.3, with some short peaks around 0.35 and a significant peak at 0.45 around 6:00 PM, as shown in Figure 6.19b. This significant peak corresponds to a period of low traffic, suggesting that balancing load is more challenging when fewer vehicles are distributed across more RSUs. Despite these fluctuations, the ARHC strategy handles the load effectively, with QoS remaining perfect throughout the entire run, making further QoS analysis unnecessary.

- **Worst-Case Scenario: Morning Traffic at Half RSU Capacity**

  In the worst-case scenario, with morning traffic and quarter RSU capacity, the Gini coefficient performs relatively well, primarily between 0.05 and 0.1, with occasional peaks around 0.15 and a brief spike above 0.175. Notably, the highest Gini spike also occurs during a period of lower traffic, similar to the best-case scenario, reinforcing the difficulty of balancing load in such conditions. However, as traffic increases, particularly beyond 60 concurrent vehicles, QoS begins to degrade, with minimum QoS dropping to around 75% and average QoS to about 94%. This suggests that the QoS drop is primarily due to localized RSU overloads, as the detailed analysis confirms that distance-based QoS remains at 100%. In contrast, the overall QoS decreases, reflecting the strain on specific RSUs with reduced capacity.



(a) Vehicles Over Time (10s Rolling Mean)

(b) Gini Scores Over Time (30s Rolling Mean)

(c) QoS Over Time (10s Rolling Mean)

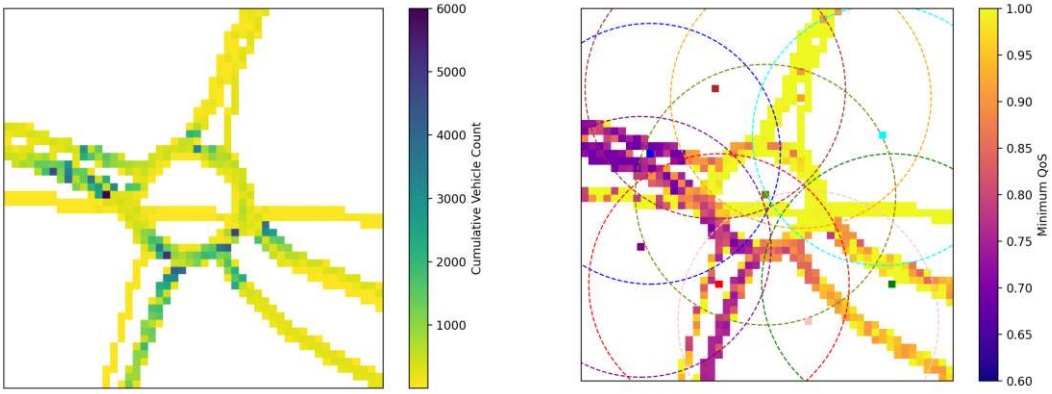
(d) Detailed QoS Over Time (10s Rolling Mean)

Figure 6.20: ARHC-01s Performance Over Time (Créteil, Evening, Dense, Quarter Capacity)

Comparing both scenarios, the ARHC strategy demonstrates effective load distribution across varying traffic conditions. Still, it encounters more significant challenges during periods of low traffic and in maintaining QoS during high-traffic situations. In the best-case scenario, the Gini coefficient rises during low traffic but remains manageable during peak periods, highlighting the strategy's capability to balance load under heavier traffic. In contrast, the worst-case scenario shows more consistent load balancing, likely due to a stronger emphasis on optimizing limited resources. However, QoS degrades when traffic exceeds certain thresholds, illustrating both the resilience and limitations of the ARHC strategy in maintaining service quality under extreme conditions.

Traffic Density and QoS

The Traffic Density Map, visualized in Figure 6.21a, illustrates the distribution of vehicles during the morning rush hour within the dense RSU configuration at quarter capacity. This scenario uses a load-sharing interval of 1 second to manage handovers and distribute computational tasks. The corresponding QoS Map, shown in Figure 6.21b, provides insight into the network's ability to maintain service quality under these conditions.



(a) Traffic Density Map (b) Minimum QoS Map

Figure 6.21: Créteil Roundabout: Traffic Density and QoS Maps (Dense Configuration, Quarter Capacity, Morning)

One key observation is the relatively high QoS maintained in the center of the roundabout and the southwestern section. These areas benefit from the coverage provided by multiple RSUs, even with each RSU operating at quarter capacity. The ARHC strategy effectively leverages this overlapping coverage, contributing to a more uniform distribution of QoS and reducing the likelihood of service quality degradation. This results in a minimum QoS of approximately 68% in these critical areas, indicating robust network performance despite the reduced capacity of individual RSUs.

Throughout the roundabout, the minimum QoS remains relatively high, reflecting the efficiency of the dense RSU deployment in handling traffic loads under the ARHC strategy. However, certain areas, such as the western entry and exit and the southern entry, exhibit lower QoS values. These regions likely experienced peak traffic during the simulation, which placed additional strain on the RSUs, leading to a slight drop in service quality. The ARHC strategy's load distribution capabilities help manage these challenges, though these areas still reflect the limits of RSU capacity under heavy load.

The eastern section of the roundabout demonstrates strong QoS, with only a few isolated spots showing notable decreases. This suggests that the dense RSU configuration, combined with the ARHC strategy, effectively manages the dynamic traffic patterns in this area, maintaining high service quality even as traffic conditions fluctuate.

In the northern part of the roundabout, where traffic is lighter, QoS remains near-perfect throughout the simulation. The low traffic volume in this area allows the RSUs to operate well within their capacity, ensuring consistently high service quality.

Overall, the dense RSU configuration with 9 RSUs at quarter capacity, supported by the ARHC strategy, effectively maintains high QoS across the roundabout. The overlapping coverage and efficient load distribution facilitated by the ARHC strategy contribute significantly to the network's strong performance, even in areas with heavier traffic.

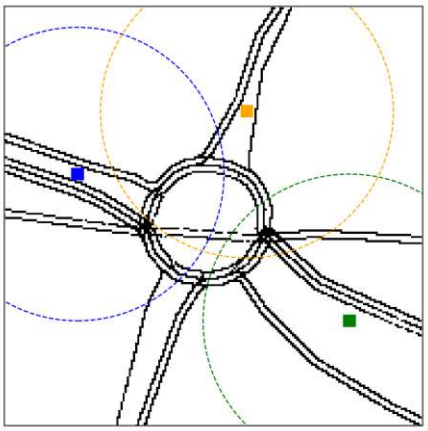


Figure 6.22: Créteil Roundabout Sparse Configuration after South RSU Failure

6.2.3 Experiment 3: Impact of RSU Failure on Strategy Performance

This experiment examines the impact of an RSU failure on the performance of the ARHC strategy and baseline approaches, focusing on key metrics such as QoS, load balancing, and handover frequency. The scenario uses the sparse 4-RSU setup from Experiment 1, with the south RSU (red) deliberately disabled to simulate a failure (see Figure 6.22). This failure scenario is tested under full and half RSU capacity settings, analyzing the morning traffic trace.

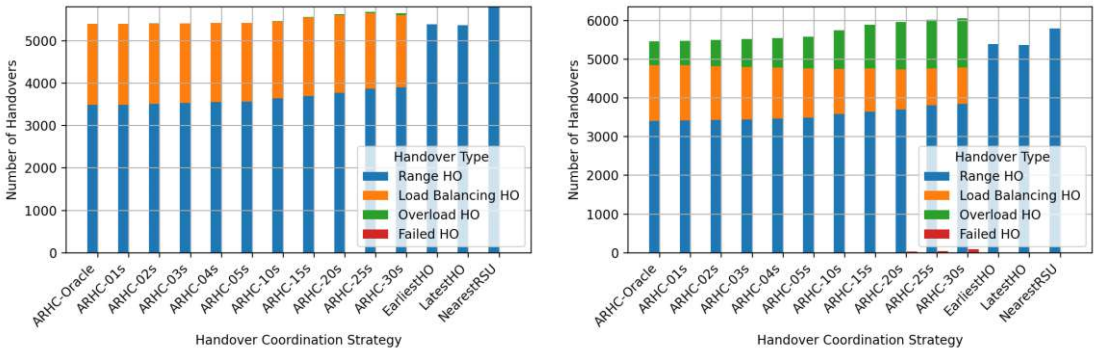
The loss of an RSU is expected to cause a noticeable drop in QoS, particularly in the affected areas. This marks the first instance where the ARHC strategy experiences significant distance-based QoS degradation. The remaining RSUs are expected to handle more load, leading to higher handover counts and potential imbalances in load distribution. This experiment underscores the importance of evaluating how robust VEC strategies are against RSU failures, which can happen due to network outages, power failures, hardware malfunctions, or physical damage.

Handover Frequency

Figure 6.23 shows handover counts during the morning trace for both full and half RSU capacities in the failure scenario.

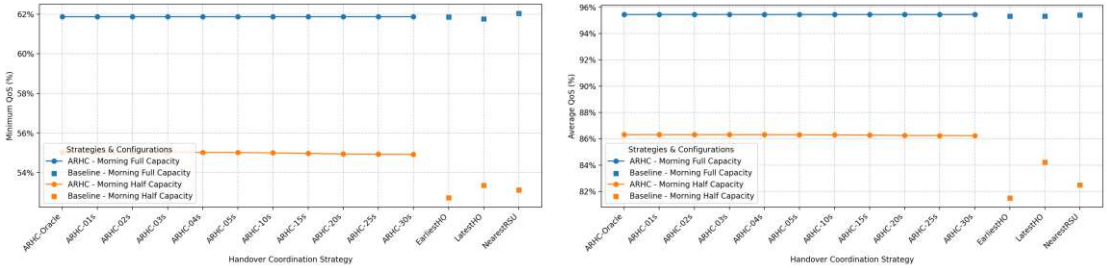
Handovers are relatively consistent across configurations at full capacity, with around 5500 total handovers. Most handovers are triggered by vehicles moving out of range in the ARHC setup, showing efficient management as vehicles pass through RSUs coverage areas. No failed handovers occur, reflecting effective load distribution despite the RSU failure.

In the half-capacity scenario, the pattern remains similar, but overload-triggered handovers appear, showing the system's efforts to balance load among the remaining RSUs. Failed handovers are minimal, even when the load-sharing intervals are high.



(a) Full RSU Capacity (b) Half RSU Capacity

Figure 6.23: Number of Handovers of Créteil Failure Configuration at Morning



(a) Minimum QoS Scores (b) Average QoS Scores

Figure 6.24: QoS Scores of Créteil Failure Configurations

These findings demonstrate the ARHC strategy's resilience in managing handovers effectively, even with a disabled RSU. However, due to the road layout, the low RSU density limits handover options, leading to a strict order of handovers. Increasing RSU density could improve flexibility.

One limitation is that ARHC only considers handovers to RSUs within a vehicle's range. If a vehicle exits a station's range and loses coverage, it won't reconnect until it enters another station's range, even if a closer RSU exists. Addressing this could improve the system's ability to handle unexpected disruptions.

Overall, the experiment highlights the ARHC strategy's ability to maintain operational integrity under challenging conditions, with opportunities for future enhancements in adaptability and coverage management.

QoS Metrics

The impact of the RSU failure on QoS reveals important insights into the resilience of the ARHC strategy compared to baseline approaches. Notably, the load-sharing interval

did not significantly affect either minimum or average QoS across different capacity configurations.

Under full capacity, the network maintained a relatively high average QoS of around 94%, even in the presence of an RSU failure (Figure 6.24b). This suggests the system was robust enough to sustain decent overall service quality. However, the minimum QoS dropped significantly to 62% (Figure 6.24a) indicating that a small group of vehicles experienced notable service degradation. This drop shows the difficulty of maintaining consistent service levels across all vehicles, especially when some are more affected by the RSU failure than others. The similar performance of the ARHC strategy and baseline approaches under full capacity conditions suggests that the high amount of available RSU capacity reduced the need for advanced load balancing, leading to comparable results.

In contrast, when the RSUs operated at half capacity, there was a noticeable decline in both minimum and average QoS. The ARHC strategy outperformed the baseline approaches, showing its effectiveness in balancing the load and preserving QoS as much as possible under constrained conditions. However, the drop in QoS can be mainly attributed to the strategy's limitation of only considering handovers to stations within range. As vehicles exited the coverage area of the overloaded western (blue) RSU, they often could not connect to another station if none was within immediate range, leading to localized service degradation. This issue was particularly severe for vehicles that exited the roundabout using the southern direction or entered from the south, where the lack of an alternative connection point further increased the load on the remaining RSUs.

Overall, while the ARHC strategy showed strong performance under full capacity, its limitations became more apparent in the half-capacity scenario, especially in dealing with RSU failures. Future improvements could enhance the handover logic to consider potential connections to nearby but out-of-range stations, increasing the system's resilience in failure scenarios.

Load Balancing Efficiency

As mentioned earlier, the load-balancing analysis shows that both capacity configurations performed similarly due to the straightforward setup and limited handover options. However, increasing the load-sharing interval led to a noticeable rise in load distribution inequality, as shown in Figure 6.25. This trend highlights the important role of frequent load-sharing in keeping load distribution balanced across RSUs.

The ARHC strategy outperformed the baseline approaches, maintaining a more balanced load distribution. Among the baseline strategies, the latest handover (HO) strategy performed the best but still fell short compared to the ARHC strategy's effectiveness in managing load imbalances. This difference is shown in Figure 6.25, where the ARHC strategy consistently outperforms the baseline methods. These findings emphasize the ARHC strategy's superior ability to optimize network resources, especially in scenarios where balanced load distribution is essential for maintaining overall system performance.

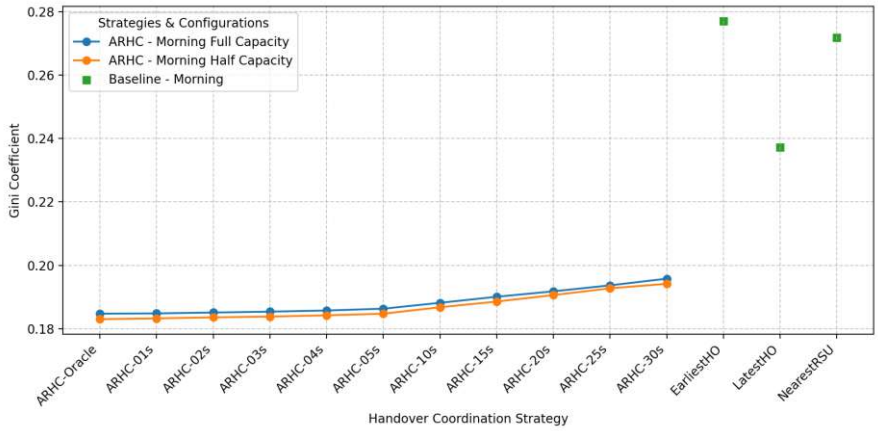


Figure 6.25: Gini Scores of Créteil Failure Configurations

Time-Based Analysis

This time-based analysis focuses on the Créteil Roundabout morning mobility trace, using the ARHC strategy with a 1-second load-sharing interval. The study uses a 10-second rolling mean for traffic and QoS metrics, while Gini scores are averaged over a 30-second window to provide a clear understanding of performance variations over time.

- **Full RSU Capacity**

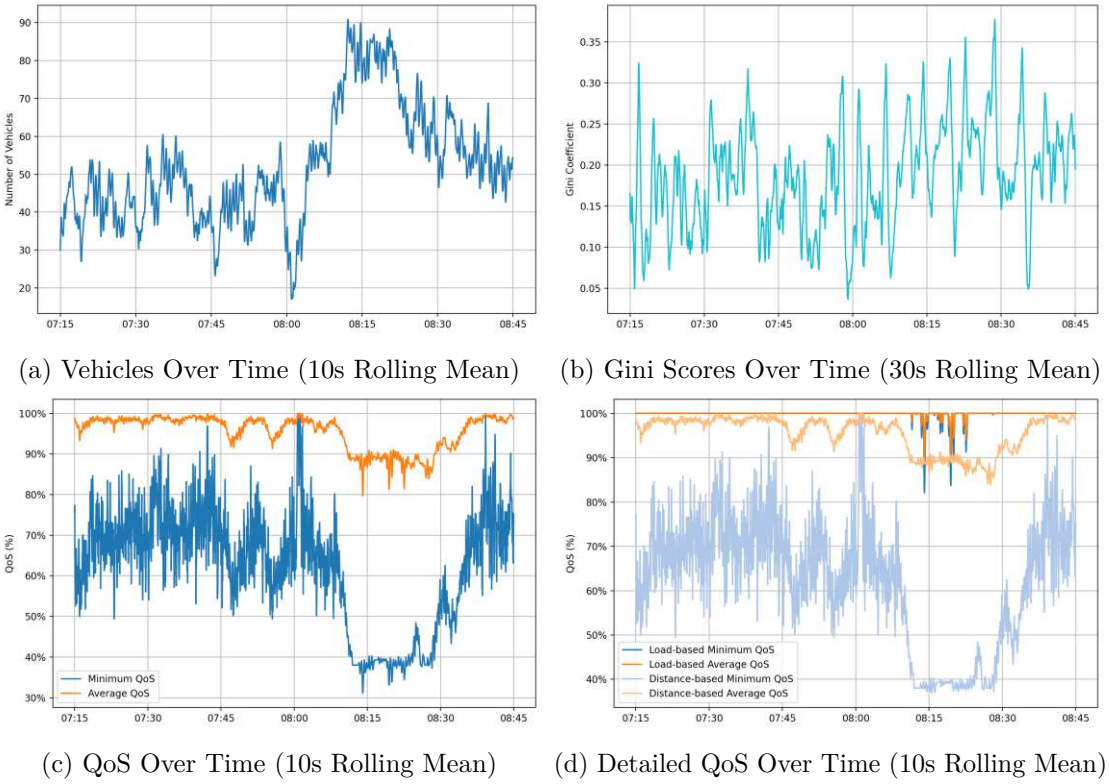
  At full RSU capacity, the Gini coefficient (Figure 6.26b) ranges mostly between 0.1 and 0.25, with occasional peaks reaching up to 0.37. Although higher traffic correlates with some increases in load imbalance, these variations are not significantly more pronounced during peak traffic compared to other times.

  The QoS metrics present a more critical picture. Both minimum and average QoS (Figure 6.26c) are stable for much of the time but drop significantly during periods of high traffic. When the number of vehicles exceeds approximately 60, the minimum QoS decreases to 30%, and the average QoS drops as low as 80%, with typical levels around 90%. Detailed analysis (Figure 6.26d) shows that these declines are mainly driven by distance-based QoS issues, as vehicles move out of the coverage range of any RSU. In contrast, minor drops in load-based QoS during peak traffic are not the leading causes of the overall QoS reduction.

- **Half RSU Capacity**

  At half RSU capacity, load distribution inequality remains similar to the full-capacity scenario, primarily due to the limited options for executing effective handovers, as visualized in Figure 6.27b.

  However, QoS metrics show greater fluctuation in Figure 6.27c. While minimum QoS is relatively stable during low traffic, it drops sharply to around 20% during



(a) Vehicles Over Time (10s Rolling Mean)

(b) Gini Scores Over Time (30s Rolling Mean)

(c) QoS Over Time (10s Rolling Mean)

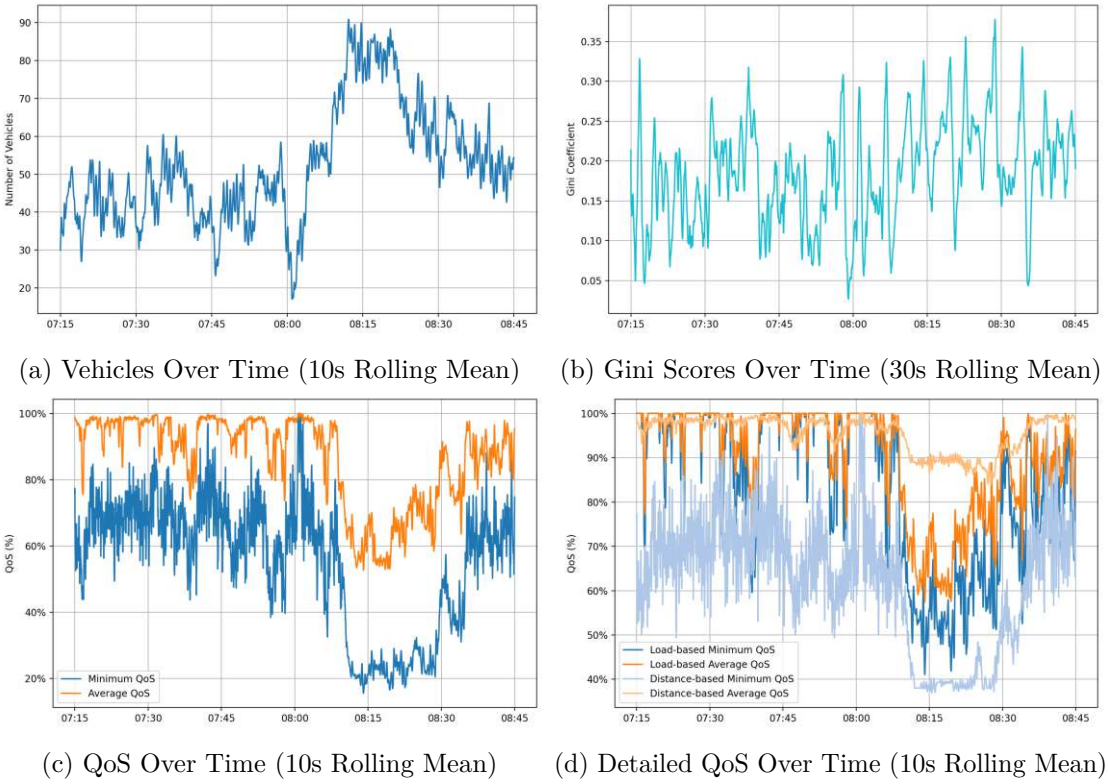
(d) Detailed QoS Over Time (10s Rolling Mean)

Figure 6.26: ARHC-01s Performance Over Time (Créteil, Morning, RSU Failure Scenario, Full Capacity)

peak traffic periods. Average QoS also declines significantly, falling to around 60% during high traffic, though it shows signs of recovery as traffic decreases.

The detailed QoS analysis at half capacity (Figure 6.27d) shows a notable impact on both load-based and distance-based QoS. The issues with distance-based QoS are consistent with those seen at full capacity, reflecting the system's challenges in maintaining coverage in certain areas. However, load-based QoS shows more significant drops, especially during peak traffic, with values falling to a minimum of 50% and an average of around 60%. This suggests a general system overload during these times, with some RSUs operating beyond their capacity and no other RSUs available to distribute the load effectively. The combination of these QoS factors results in significantly lower overall minimum and average QoS during peak traffic periods.

Comparing the two scenarios highlights that while load distribution inequality remains consistent due to the limited handover options, QoS is much more affected at half capacity. When operating with reduced capacity, the system struggles to maintain satisfactory QoS levels during peak traffic, mainly because of the lack of available RSUs to manage the



(a) Vehicles Over Time (10s Rolling Mean)

(b) Gini Scores Over Time (30s Rolling Mean)

(c) QoS Over Time (10s Rolling Mean)

(d) Detailed QoS Over Time (10s Rolling Mean)

Figure 6.27: ARHC-01s Performance Over Time (Créteil, Morning, RSU Failure Scenario, Half Capacity)

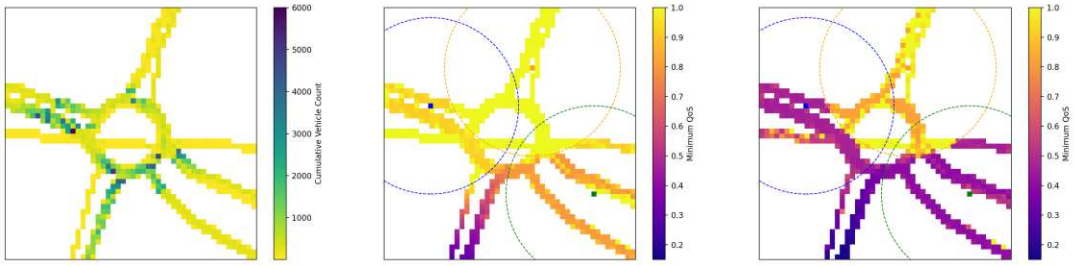
increased load effectively. This analysis underscores the critical importance of maintaining adequate RSU capacity to ensure stable QoS, particularly during periods of high traffic.

Traffic Density and QoS

The QoS maps clearly illustrate the regional impact of reduced coverage, particularly under conditions of RSU failure, and how this affects the minimum QoS across different areas.

In the full RSU configuration (Figure 6.28b), the southern area, where RSU coverage is absent due to failure, shows the lowest minimum QoS values. This drop is most pronounced in the areas furthest from the remaining operational RSUs, confirming the expected exponential decrease in QoS as vehicles move out of range, as detailed in Section 4.7.1. Additionally, the eastern RSU (green) exhibits minor decreases in minimum QoS, likely caused by the brief load-based QoS drops identified earlier.

The situation worsens in the half-capacity scenario (Figure 6.28c). The southern region continues to suffer from significant distance-related QoS degradation due to the lack of coverage. Moreover, the reduced RSU capacity has a pronounced negative effect on the



(a) Traffic Density Map (b) Min. QoS Map (Full Cap.) (c) Min. QoS Map (Half Cap.)

Figure 6.28: Créteil Roundabout: Traffic Density and QoS Maps (Failure Configuration, Morning)

western (blue) and eastern (green) RSUs. The absence of additional RSUs to redistribute the load effectively increases the problem, with the northern (yellow) RSU too distant to provide meaningful support without further compromising distance-based QoS.

The current design of the ARHC strategy, which does not account for handovers to out-of-range stations, contributes to this issue. While such handovers might sacrifice distance-based QoS for a few vehicles, they could potentially alleviate the load on overloaded RSUs, offering a trade-off that might improve overall network performance. This limitation highlights a potential area for future improvement of the ARHC strategy, where considering handovers to more distant RSUs could help balance the load more effectively under failure conditions.

6.3 Cross-Experiment Analysis of Normal Operations

In this section, we compare the performance of the explored handover strategies, particularly the ARHC strategy, across the sparse and dense RSU configurations of the Créteil roundabout, previously detailed in Section 6.2.1 and Section 6.2.2. This analysis focuses exclusively on normal operations under different RSU configurations and does not include the RSU failure scenario discussed in Experiment 3. To streamline the analysis, we focus on the morning mobility trace, given its higher average and peak vehicle counts compared to the evening trace. By integrating the results from both configurations, this analysis offers enhanced visualizations and insights, highlighting the key differences and performance variations between the sparse and dense setups.

Impact of Traffic Density and Network Congestion

One critical aspect influencing the performance of the ARHC strategy across these experiments is the varying traffic density between the configurations. The sparse configuration, with fewer RSUs, encounters more significant challenges during high-traffic periods, particularly in maintaining optimal QoS. This is primarily due to the limited resources available to handle the increased load, leading to more pronounced drops in

QoS and a higher frequency of handovers. The time-based analysis of the morning traffic scenario highlighted that as vehicle counts approached and exceeded 60, QoS began to degrade sharply in the sparse setup, illustrating its vulnerability to traffic congestion.

In contrast, the dense configuration, featuring a greater number of RSUs, benefits from enhanced coverage and more opportunities for load balancing. This setup allows the ARHC strategy to distribute the computational load more effectively, even during peak traffic periods. The dense configuration's overlapping coverage proved particularly effective during periods of high congestion, as shown in the time-based analysis, where it maintained higher QoS levels despite similar traffic increases.

Handover Frequency Comparison

The number of handovers observed in the sparse and dense configurations reveals distinct patterns, driven primarily by the number of RSUs and the load-balancing strategies employed.

In the sparse configuration, shown in Figure 6.6, the overall number of handovers is significantly lower, reflecting the fewer RSUs available. Overload-triggered handovers in this setup occur at half capacity, and while they are relatively infrequent, their impact is more pronounced due to the limited RSU resources. The time-based analysis further emphasized that handovers became more critical during peak traffic in maintaining service, particularly when vehicle counts increased rapidly.

Conversely, the dense configuration, illustrated in Figure 6.14, sees a much higher number of handovers, driven by more aggressive load balancing needed for the larger number of RSUs. Overload-triggered handovers are delayed until RSU capacity drops to a quarter, thanks to the additional RSUs. However, when they occur, these handovers are more frequent in the sparse configuration, reflecting the strain on fewer RSUs. As observed in the time-based analysis, the dense setup managed these handovers more smoothly, maintaining network stability even during significant traffic increases.

Failed handovers appear only in the most resource-constrained scenarios: At half capacity in the sparse setup and quarter capacity in the dense setup. The sparse configuration experiences minimal failed handovers, primarily at high load-sharing intervals, while the dense configuration shows a significant increase in failed handovers at quarter capacity, indicating challenges in managing high traffic loads despite more RSUs.

When comparing ARHC to baseline strategies, the sparse configuration shows that ARHC remains competitive, maintaining handover efficiency even under reduced capacity and high load-sharing intervals. However, ARHC results in substantially more handovers than baseline strategies in the dense configuration, significantly as RSU capacity decreases and load-sharing intervals increase. This suggests that while ARHC is effective in sparse environments, its application in dense configurations may lead to an increased frequency of handovers, potentially impacting overall network performance and efficiency. On the other hand, this allows ARHC to distribute the computational load more evenly.

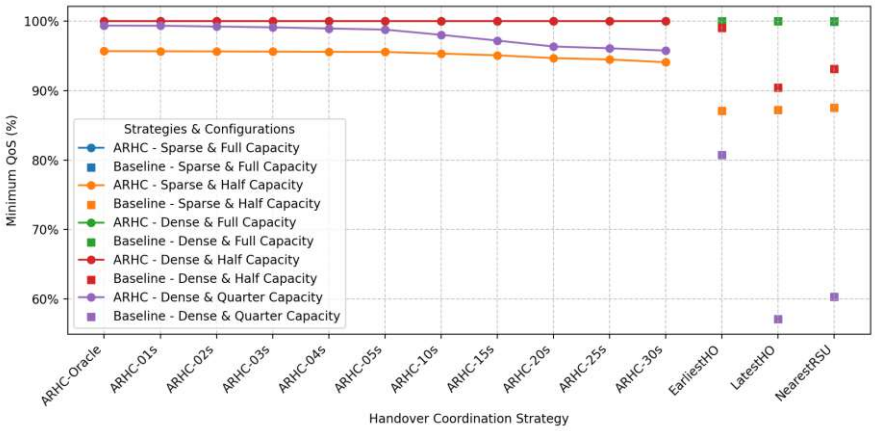


Figure 6.29: Minimum QoS Scores of Créteil Sparse & Dense Morning Configurations

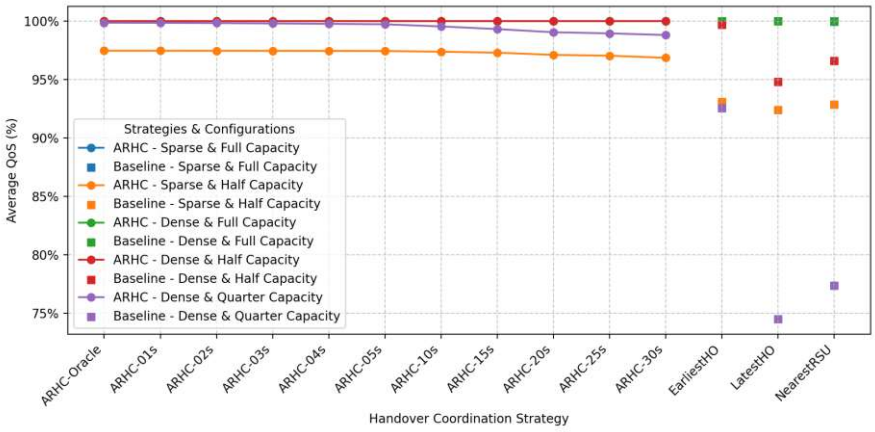


Figure 6.30: Average QoS Scores of Créteil Sparse & Dense Morning Configurations

QoS Comparison

Figures 6.29 and 6.30 present the minimum and average QoS scores for the sparse and dense configurations during the morning trace.

For ARHC, optimal minimum and average QoS are consistently achieved in the sparse & full capacity, dense & full capacity, and dense & half RSU capacity configurations across all load-sharing intervals. However, performance differences become apparent as RSU capacity decreases. The dense & quarter capacity configuration experiences a slight decline in QoS, while the sparse & half capacity configuration shows an even more noticeable reduction. Higher load-sharing intervals affect the dense & quarter configuration more adversely, with a more significant QoS drop than the sparse & half-capacity setup. This pattern holds consistently across both minimum and average QoS metrics, exhibiting similar trends.

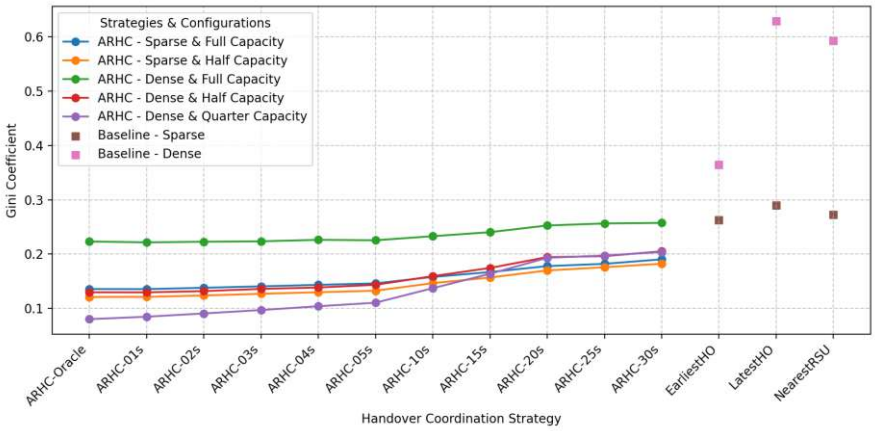


Figure 6.31: Gini Scores of Créteil Sparse & Dense Morning Configurations

Baseline strategies generally fall short of ARHC in QoS performance, except at full capacity, where they achieve optimal scores. All baseline strategies produce similar QoS outcomes in the sparse configuration. However, in the dense configuration, the earliest HO strategy significantly outperforms the other baseline approaches, particularly in scenarios with more RSUs.

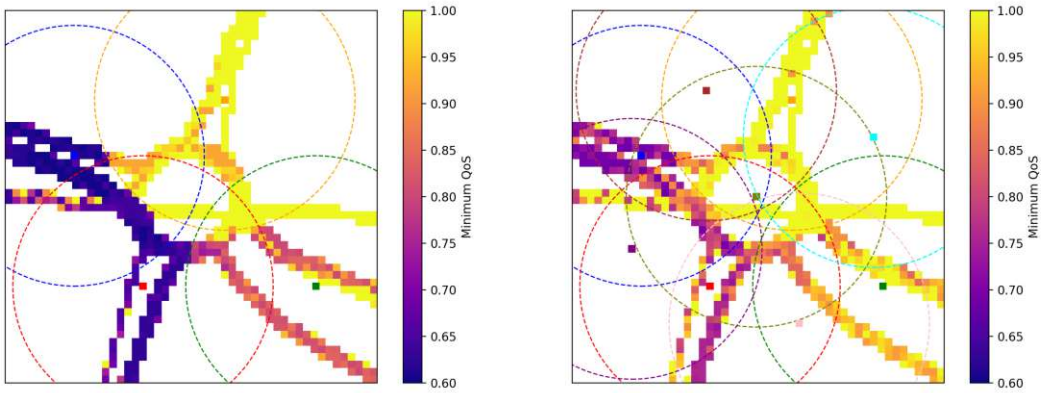
This performance boost highlights a vital advantage of the earliest HO strategy in dense environments, though it comes with the trade-off of a higher number of handovers. These results emphasize the importance of considering handover frequency and QoS in evaluating the effectiveness of different strategies under varying network conditions.

Load Balancing Comparison

Figure 6.31 presents the Gini scores for load balancing across the sparse and dense configurations during the morning trace.

Among the ARHC configurations, the dense, quarter RSU capacity setup initially achieves the best load-balancing performance. However, as the load-sharing interval increases, its performance deteriorates the most, eventually becoming worse than all other ARHC configurations except for the dense full-capacity setup. The dense & full-capacity configuration consistently performs the worst load balancing among all ARHC setups.

This pattern can be explained by the behavior of RSUs under reduced capacity. When RSUs operate with lower capacity, they are more likely to reach their load limits. Since all RSUs have the same maximum capacity, they tend to carry similar loads once fully utilized, leading to a more even distribution across the network, as explained in Section 4.7.1. In these scenarios, the ARHC strategy frequently balances load to prevent overloads, naturally resulting in a lower Gini index. The time-based analysis also highlighted that load imbalances in the sparse configuration became more pronounced during traffic peaks, contributing to higher Gini scores and more significant strain on individual RSUs.



(a) Sparse & Half Capacity Configuration (b) Dense & Quarter Capacity Configuration

Figure 6.32: Créteil Roundabout: Minimum QoS Maps (Morning)

The other ARHC configurations exhibit similar performance, including sparse & full capacity and sparse & half capacity, maintaining relatively balanced load distribution as load-sharing intervals increase. These configurations are less affected by the lengthening of load-sharing intervals compared to the dense full and dense quarter RSU capacity setups.

Baseline strategies, as expected, perform worse in load balancing than ARHC. The sparse baseline configurations show similar performance to each other and outperform the baseline strategies in the dense configuration. Among the dense baseline strategies, the earliest HO strategy stands out by achieving better load balancing than the other baseline approaches. However, even this strategy falls short compared to the ARHC configurations in terms of maintaining optimal load balance.

Minimum QoS Map Comparison

The comparison between the minimum QoS maps for the sparse and dense RSU configurations, as shown in Figure 6.32, highlights the effectiveness of the proposed ARHC strategy in managing handover coordination and maintaining service quality across different network setups.

In the dense configuration with 9 RSUs at quarter capacity, the ARHC strategy shows significant improvement in QoS, particularly in the center of the roundabout and the southwestern section. This is due to the strategy's effective use of overlapping RSUs coverage, leading to smoother transitions and reducing the sharp drops in QoS seen in the sparse configuration. The time-based analysis demonstrated that these areas benefited from more consistent QoS during peak traffic periods, further reinforcing the advantage of the dense configuration.

The ARHC strategy maintains a minimum QoS of around 68%, even under reduced RSU capacity, showcasing its ability to optimize network resources and prevent severe service

degradation. While areas like the western entry and exit, and the southern entry still face lower QoS, the strategy's load distribution capabilities lower the impact of traffic peaks in the dense setup.

In the eastern section, the ARHC strategy continues to maintain near-perfect QoS, adapting well to dynamic traffic conditions. With lighter traffic, the northern part also benefits from robust handover coordination, achieving near-perfect QoS throughout.

Overall, the ARHC strategy effectively manages handovers and sustains high QoS levels in the dense configuration, even with reduced RSU capacity. Its ability to leverage more RSUs to smooth transitions and evenly distribute load makes it a robust solution for vehicular edge computing environments.

6.4 Discussion

6.4.1 Comparative Insights from Experiments 1 and 2

The cross-experiment analysis provides a holistic evaluation of the Créteil Roundabout experiments, focusing on the comparative performance of the ARHC strategy across sparse and dense RSU configurations at normal operations. This section synthesizes the key findings from both experimental setups, highlighting the strengths and limitations of ARHC under varying network conditions.

1. **Effectiveness of ARHC in Managing Handovers**
   In sparse and dense configurations, ARHC demonstrates a strong ability to manage handovers efficiently, particularly under full-capacity conditions. The strategy's decentralized approach, leveraging RSU-based decision-making, enables it to control handover processes, minimizing unnecessary disruptions and maintaining network stability. However, as RSU capacity decreases, the increased complexity in dense configurations introduces more frequent handovers, driven primarily by the need for aggressive load balancing. This was particularly evident during periods of high traffic, where the rise in vehicle numbers required more frequent and critical handovers in the sparse setup to maintain service continuity. This indicates that while ARHC is highly effective in environments with limited infrastructure (such as the sparse configuration), its application in denser networks requires careful management to prevent excessive handover frequency, which could otherwise lead to increased network overhead and reduced QoS.

2. **QoS Maintenance and Load Balancing Efficiency**
   The performance of ARHC in maintaining QoS is significantly influenced by the total network capacity and the distribution of RSUs. The total capacity is inherently lower in the sparse configuration due to the smaller number of RSUs (4 compared to 9 in the dense configuration). This limitation becomes more pronounced as RSU capacity is reduced, making it difficult for ARHC to maintain high QoS levels, especially in high-traffic scenarios.

Interestingly, load balancing tends to improve in reduced-capacity scenarios, as indicated by the lower Gini index. This happens because all RSUs, having the same capacity, reach similar load levels when fully utilized. The ARHC strategy responds by balancing the load more frequently to prevent overloads, leading to a more even distribution of the load across the network. This behavior highlights a key relationship between RSU capacity and load distribution: Reduced capacity can lead to better load balance, though it may come at the cost of overall QoS if the system's total capacity is insufficient.

In contrast, the dense configuration benefits from a higher total capacity and more overlapping RSU coverage, which enhances the system's ability to manage peak loads and maintain higher QoS. Even as individual RSU capacity decreases, the dense setup's greater number of RSUs allows for better load distribution, helping to mitigate the impact of traffic surges.

3. **Implications for Deploying ARHC**
   The findings from these experiments highlight several key considerations for deploying the ARHC strategy in vehicular edge computing environments. The effectiveness of load balancing is closely tied to the available RSU capacity. In reduced-capacity scenarios, RSUs tend to reach similar load levels once fully utilized, which prompts the ARHC strategy to engage in more frequent load balancing. This results in a more even distribution of the load across the network. However, sufficient RSU capacity is essential to prevent potential QoS degradation in high-traffic areas. Therefore, strategically deploying RSUs to ensure optimal coverage and capacity is crucial for maximizing the performance of the ARHC strategy, particularly in environments with varying traffic demands.

The performance of ARHC across sparse and dense RSU configurations reveals that the strategy is robust and effective. Still, its success is highly dependent on the network's total capacity and the distribution of RSUs. With its higher capacity and overlapping RSU coverage, the dense configuration provides a more favorable environment for ARHC, enabling better management of peak loads and higher overall QoS. In contrast, the sparse configuration's limited capacity presents challenges that require careful management to avoid significant QoS degradation. These findings underscore the importance of optimizing network topology and RSU distribution to maximize the effectiveness of ARHC in diverse VEC environments.

6.4.2 Limitations

While the results demonstrate the efficacy of ARHC in various scenarios, several limitations must be acknowledged. Firstly, the scope of the simulation is confined to a relatively small urban area, specifically the Créteil Roundabout. While this setting provides valuable insights, it may not fully capture the challenges encountered in more extensive or diverse vehicular networks, such as highways or rural areas. This limitation

suggests that further research is necessary to assess ARHC's performance across various environments.

Secondly, the simulation assumes consistent RSU availability and reliable communication, overlooking potential real-world issues such as RSU failures or data transmission errors. The findings from the failure scenario highlight ARHC's dependence on RSU availability, where the loss of an RSU led to significant QoS degradation, particularly in sparse configurations where redundancy is minimal. This reliance on uninterrupted RSU operation underscores the importance of enhancing the strategy's robustness to handle such disruptions.

Thirdly, the simplified load-offloading model used in the simulations focuses solely on computational load, measured in GFLOPS, without accounting for other factors like memory usage or specific task requirements. While this model is adequate for initial evaluations, it does not fully represent the complexity of real-world vehicular tasks, which often involve multiple resource constraints. Future research should consider a more comprehensive model that includes these additional factors.

6.4.3 Strengths and Shortcomings

This section provides a comprehensive overview of the strengths and shortcomings of the proposed ARHC strategy, as observed through the experiments conducted at the Créteil Roundabout. This analysis highlights the effectiveness of ARHC and identifies areas for potential improvement and future research.

Strengths of the ARHC strategy

1. **Scalability and Adaptability**
   One of the most significant strengths of the ARHC strategy is its scalability across different network environments. The approach has proven effective in both sparse and dense RSU configurations, demonstrating its ability to adapt to varying network density and capacity levels. This scalability ensures that ARHC can be applied in diverse VEC scenarios, ranging from small urban areas with limited RSUs to larger, more complex networks with high RSU density.

2. **Effective QoS Maintenance**
   The ARHC strategy consistently maintains high QoS, particularly under full RSU capacity conditions. Even in scenarios where RSU capacity is reduced, ARHC outperforms baseline strategies by effectively managing handovers and balancing computational loads. This ability to sustain high QoS under varying traffic conditions highlights ARHC's robustness and potential for real-world deployment in vehicular networks.

   However, when faced with an RSU failure, the strategy's effectiveness is significantly compromised. The reduction in available network infrastructure led to noticeable

QoS degradation, particularly in areas with sparse RSU coverage. This suggests the need for further enhancements to better handle such scenarios.

3. **Balanced Load Distribution**
   The Gini coefficient analysis across multiple experiments demonstrates that ARHC achieves a more balanced distribution of computational loads among RSUs, especially in resource-constrained scenarios. By preventing RSU overloads and ensuring that no RSU becomes a bottleneck, ARHC enhances the overall stability and efficiency of the network. This balance is crucial in maintaining consistent service levels for all vehicles within the network, particularly during periods of high traffic.

   In the event of network disruptions, such as an RSU failure, maintaining this balance becomes even more critical. The absence of an RSU exacerbates load imbalances, particularly when the strategy does not account for handovers to out-of-range stations, which could mitigate these effects.

4. **Robust Handover Coordination**
   ARHC performs well in coordinating handovers, minimizing unnecessary handovers, and preventing failures even in scenarios with reduced RSU capacity. This robust coordination is essential for maintaining uninterrupted service as vehicles move through different RSU coverage areas, ensuring that computational tasks are seamlessly offloaded without compromising QoS.

   Nonetheless, in scenarios where the network infrastructure is compromised, such as during an RSU failure, the current handover logic of ARHC could be improved by considering connections to nearby but out-of-range stations. This could help sustain network stability and service quality even under challenging conditions.

Shortcomings of the ARHC strategy

1. **Dependence on RSU Capacity**
   The effectiveness of ARHC highly depends on the available RSU capacity. In scenarios where RSU capacity is significantly reduced, ARHC's ability to maintain high QoS and balanced load distribution diminishes. Over time, it was observed that in sparse configurations, rapid increases in traffic could quickly lead to QoS drops due to limited RSUs, highlighting the strategy's dependence on sufficient capacity. This limitation was particularly evident when the network faced an RSU failure, further underscoring the strategy's vulnerability in low-capacity scenarios.

2. **Manual Configuration of RSUs and Parameters**
   The manual configuration of RSU placement, capacity, and load-sharing intervals introduces potential for human error and may affect the reproducibility of the simulation results. An algorithmic approach to RSU placement and parameter configuration could enhance the precision and reliability of the simulations, leading to more accurate evaluations of ARHC's performance in various network scenarios.

The ARHC strategy demonstrates considerable strengths, including scalability, robust handover coordination, and effective QoS maintenance, making it a promising solution for VEC. However, the shortcomings identified in both normal operations and failure scenarios highlight areas where further research and development are needed. Addressing these limitations, particularly in expanding the simulation scope, optimizing ARHC for low-capacity scenarios, and incorporating edge cases and failure scenarios, will enhance the strategy's applicability and robustness.

As we conclude this discussion, it's clear that ARHC offers valuable insights and solutions for VEC. However, the limitations identified also point to areas where further work is needed. The following conclusion will summarize the key findings and consider how these insights contribute to the broader field of VEC while briefly outlining future research directions.

CHAPTER 7

Conclusion and Future Work

This thesis addressed the challenge of handover coordination in VEC environments by introducing the ARHC strategy. Through the innovative integration of MASs within an RSU-based communication framework, this research demonstrates significant improvements in network performance, resource utilization, and QoS in vehicular networks.

The ARHC strategy innovatively decentralizes decision-making, shifting control from centralized systems or individual vehicles to RSUs within the fog computing layer. This approach enhances system reliability, scalability, and responsiveness, crucial for dynamic VEC environments where low latency and high reliability are essential. By efficiently managing computational loads and minimizing unnecessary handovers, ARHC ensures a balanced distribution of resources across RSUs while maintaining high QoS even under varying traffic conditions.

One of the key contributions of this thesis is the development of a custom simulation environment tailored for evaluating MAS-based handover strategies in VEC systems. This environment facilitated comprehensive testing of the ARHC strategy across various realistic scenarios, providing detailed insights into its effectiveness compared to traditional approaches. The ability to extract and analyze critical performance metrics, such as successful and failed handovers, load distribution, and QoS, was essential for a robust evaluation.

This work has broader implications for applying MASs in VEC systems. Distributed and collaborative decision-making processes can significantly enhance network adaptability and resilience. The RSU-based approach introduced in this thesis represents a paradigm shift in handover coordination, with decisions made closer to the network edge, thereby reducing latency and communication overhead.

In conclusion, the ARHC strategy and the accompanying simulation environment developed in this research provide a solid foundation for improving vehicular networks'

efficiency, reliability, and scalability. While the results validate the potential of the ARHC strategy, they also highlight areas for further research and development.

7.0.1 Future Work

To further enhance the ARHC strategy, the following areas could be explored:

1. **Expanded Simulation Scenarios:** Expand the simulation to assess ARHC's scalability and performance in more diverse environments, such as highways, multi-lane intersections, and urban areas.
2. **Robustness in Adverse Conditions:** Test ARHC's resilience by introducing more comprehensive failure scenarios like RSUs outages and data transmission errors, ensuring reliable performance under challenging conditions.
3. **Optimization for Constrained Resources:** Refine ARHC to maintain high QoS and effective load balancing even in low-capacity environments with limited RSU resources.
4. **Advanced Load Management:** Enhance the load-offloading model by incorporating additional factors such as memory usage and bandwidth, better reflecting the complexity of real-world vehicular tasks.
5. **Adaptive and Efficient Handover:** Investigate dynamic handover management strategies, including multi-RSU coordination and real-time adjustments, to reduce handover frequency and improve overall network efficiency.
6. **Proactive Resource Allocation:** Integrate trend prediction and historical analysis to enable ARHC to anticipate and manage network load more effectively, optimizing handover decisions and resource allocations.
7. **Machine Learning for Predictive Decision-Making:** Implement ML models to predict network load development based on historical data and to forecast vehicle trajectories, enabling more informed and proactive handover coordination.

This thesis contributes a meaningful step forward in evolving decentralized, agent-based approaches to handover coordination in VEC, with implications that resonate across ITS and urban connectivity.

Overview of Generative AI Tools Used

ChatGPT

Purpose of Use

- **Text Enhancement**: ChatGPT was employed as a supplementary tool to assist in refining the language of content I independently created. It aided in rephrasing sentences, enhancing clarity and coherence, and ensuring stylistic consistency. My academic input entirely shaped the original ideas, content, and overall structure.

- **Script Assistance**: ChatGPT supported generating Python scripts for visualizing experimental results using Matplotlib. These scripts were based on experimental data that I designed and analyzed, with the overall design and implementation independently developed by me.

Extent of Use

- All outputs from ChatGPT were meticulously reviewed and underwent substantial changes to ensure that the final content and scripts accurately reflected my academic objectives and personal contribution.

Grammarly AI Text Assistant

Purpose of Use

- **Grammar and Style Improvement**: Grammarly's AI text assistant was utilized to enhance grammatical accuracy and stylistic consistency throughout the thesis. It provided suggestions to improve sentence structure, punctuation, and overall readability.

Extent of Use

- Grammarly's suggestions were a preliminary step in the refinement process and were significantly adjusted to ensure the final text accurately reflected my original ideas and writing style.

List of Figures

List of Tables

Acronyms

AD Autonomous Driving. xiii, 1, 7, 10–12, 23, 27, 29, 30, 40, 48, 56, 59, 63–65, 113

ADAS Advanced Driver Assistance Systems. xiii, 1

AI Artifical Intelligence. 18

AP Access Point. 12, 22

AR Augmented Reality. 7, 11, 12

ARHC Agent-based RSU Handover Coordination. xiii, 2–4, 27–30, 34, 37–39, 44–49, 51, 53–59, 61, 63, 64, 76–85, 87–108, 111, 112

CAM Cooperative Awareness Message. 33, 34, 36, 37, 46, 50, 51, 56

CMAB Contextual Multi-Armed Bandit. 19

DAI Distributed Artificial Intelligence. 13

DQN Deep Q-Network. 18

DRL Deep Reinforcement Learning. 18, 19

DSRC Dedicated Short-Range Communication. 12

EC Edge Computing. 1, 4–7, 10–12, 113

ETSI European Telecommunications Standards Institute. 34, 50

FLOP Floating Point Operations. 56, 57, 64, 65

FLOPS Floating Point Operations Per Second. 57, 64–66, 71–73, 104

FMF Follow Me Fog. 21, 22

GPU Graphics Processing Unit. 65, 72

GRA Grey Relational Analysis. 22

HHO Hard Handover. 8, 21

HO Handover. 74, 77, 78, 80, 84, 85, 87, 93, 100, 101

I2I Infrastructure-to-Infrastructure. 32, 36, 37

ILP Integer Linear Programming. 18, 19

IoE Internet of Everything. 5

IoT Internet of Things. 12, 20, 21, 24

IoV Internet of Vehicles. 19

ITS Intelligent Transportation Systems. xiii, 23, 34, 57, 68, 108

MAS Multi-Agent System. xiii, 2–5, 13–17, 19, 20, 22–27, 34, 38–40, 44–48, 56, 107

MCDM Multi-Criteria Decision-Making. 22, 24, 35, 53

MDP Markov Decision Process. 18

MDRLHA Multi-agent Deep Reinforcement Learning-based Hungarian Algorithm. 17

MEC Multi-Access Edge Computing. 1, 2, 5, 7–12, 18–21, 31, 32, 48, 111, 113

MEO MEC Orchestrator. 21

ML Machine Learning. 16, 108

MN Mobile Node. 22

OBU On-board Unit. 11, 12, 64

QoS Quality of Service. xiii, 1–3, 8, 9, 12, 14, 21, 25, 27–30, 35–45, 47, 50, 53–55, 60, 61, 63, 66, 67, 71–76, 78, 79, 81–108, 111, 112

RL Reinforcement Learning. 17

RS RoadSide. 32–38, 41, 42, 46, 48–51, 53–55, 57, 60, 64, 73, 74, 76, 111, 113

RSS Received Signal Strength. 8, 21, 22, 29, 32, 38

RSU Road Side Unit. xiii, 1–3, 11, 12, 19, 20, 23, 27–48, 50–55, 57, 59–61, 63–67, 71–74, 76–86, 88–105, 107, 108, 111, 112

SAW Simple Additive Weighting. 24

SDN Software-defined Networking. 19, 21

SHO Soft Handover. 8, 21, 22, 38

SNR Signal-to-Noise Ratio. 43

SUMO Simulation of Urban MObility. 20

TESAT Total Edge Servers Allocation Time. 18

TESU Total Edge Servers Utilization. 18

UE User Equipment. 10

V-FN Vehicle-Based Fog Node. 19

V2I Vehicle-to-Infrastructure. 11, 12, 20, 32

V2V Vehicle-to-Vehicle. 20, 32

V2X Vehicle-to-Everything. 12, 24

VE In-Vehicle. 32–35, 37, 38, 41, 46, 48–51, 55, 57, 60, 111

VEC Vehicular Edge Computing. xiii, 1–5, 10–13, 17–20, 22–30, 38, 39, 41, 42, 45–49, 51, 55–58, 61, 63, 64, 71, 91, 103, 104, 106–108, 113

VHO Vertical Handover. 22

VM Virtual Machine. 9

VMEC Vehicle-Assisted Mobile Edge Computing. 17

VNF Virtual Network Function. 18

VR Virtual Reality. 7, 11, 12

Bibliography

[1] Atiq Ahmed, Leïla Merghem-Boulahia, and Dominique Gaïti. Cooperative agent based vertical handover scheme for heterogeneous networks. In *2010 Sixth Advanced International Conference on Telecommunications*, pages 410–415, 2010.

[2] Md. Zahangir Alam and Abbas Jamalipour. Multi-agent drl-based hungarian algorithm (madrlha) for task offloading in multi-access edge computing internet of vehicles (iovs). *IEEE Transactions on Wireless Communications*, 21(9):7641–7652, 2022.

[3] Syed Danial Ali Shah, Mark A Gregory, and Shuo Li. A distributed control plane architecture for handover management in mec-enabled vehicular networks. In *2021 31st International Telecommunication Networks and Applications Conference (ITNAC)*, pages 188–191, 2021.

[4] Shaimaa R. Alkaabi, Mark A. Gregory, and Shuo Li. Multi-access edge computing handover strategies, management, and challenges: A review. *IEEE Access*, 12:4660–4673, 2024.

[5] P. G. Balaji and D. Srinivasan. *An Introduction to Multi-Agent Systems*, pages 1–27. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[6] Wei Bao, Dong Yuan, Zhengjie Yang, Shen Wang, Wei Li, Bing Bing Zhou, and Albert Y. Zomaya. Follow me fog: Toward seamless handover timing schemes in a fog computing environment. *IEEE Communications Magazine*, 55(11):72–78, 2017.

[7] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In *2009 International Conference on High Performance Computing & Simulation*, pages 1–11, 2009.

[8] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. An overview on edge computing research. *IEEE Access*, 8:85714–85728, 2020.

[9] Florent Carlier, Virginie Fresse, Jean-Paul Jamont, Arnaud Rosay, and Loïc Pallardy. A multi-edge-agent system approach for sharing heterogeneous computing resources. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pages 1–6, 2020.

[10] ETSI. Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. Technical Report EN 302 637-2 V1.3.2 (2014-11), ETSI, December 2014.

[11] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, jun 2003.

[12] Fadoua Fellir, Adnane El Attar, Khalid Nafil, and Lawrence Chung. A multi-agent based model for task scheduling in cloud-fog computing platform. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pages 377–382, 2020.

[13] Emmanuelle Grislin-Le Strugeon, Hamza Ouarnoughi, and Smail Niar. A multi-agent approach for vehicle-to-fog fair computation offloading. In *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8, 2020.

[14] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar 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*, 47(9):1275–1296, 2017.

[15] Celia Gutiérrez, Iván García-Magariño, and Rubén Fuentes-Fernández. Detection of undesirable communication patterns in multi-agent systems. *Engineering Applications of Artificial Intelligence*, 24(1):103–116, 2011.

[16] Dominik Henneke, Mohammad Elattar, and Jürgen Jasperneite. Communication patterns for cyber-physical systems. In *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–4, 2015.

[17] Xiaoge Huang, Ke Xu, Chenbin Lai, Qianbin Chen, and Jie Zhang. Energy-efficient offloading decision-making for mobile edge computing in vehicular networks. *EURASIP Journal on Wireless Communications and Networking*, 2020(1):35, February 2020.

[18] Hatem Ibn-Khedher, Mohammed Laroui, Mouna Ben Mabrouk, Hassine Moungla, Hossam Afifi, Alberto Nai Oleari, and Ahmed E. Kamal. Edge computing assisted autonomous driving using artificial intelligence. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*, pages 254–259, 2021.

[19] Ying Ju, Yuchao Chen, Zhiwei Cao, Lei Liu, Qingqi Pei, Ming Xiao, Kaoru Ota, Mianxiong Dong, and Victor C. M. Leung. Joint secure offloading and resource allocation for vehicular edge computing network: A multi-agent deep reinforcement learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 24(5):5555–5569, 2023.

[20] Jackie Kazil, David Masad, and Andrew Crooks. Utilizing python for agent-based modeling: The mesa framework. In Robert Thomson, Halil Bisgin, Christopher

Dancy, Ayaz Hyder, and Muhammad Hussain, editors, *Social, Cultural, and Behavioral Modeling*, pages 308–317, Cham, 2020. Springer International Publishing.

[21] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235, 2019.

[22] Marie-ange Lebre, Frederic Le Mouel, and Eric Menard. On the importance of real data for microscopic urban vehicular mobility trace. In *2015 14th International Conference on ITS Telecommunications (ITST)*, pages 22–26, 2015.

[23] Yun Li, Bin Cao, and Chonggang Wang. Handover schemes in heterogeneous lte networks: challenges and opportunities. *IEEE Wireless Communications*, 23(2):112–117, 2016.

[24] Bing Lin, Kai Lin, Changhang Lin, Yu Lu, Ziqing Huang, and Xinwei Chen. Computation offloading strategy based on deep reinforcement learning for connected and autonomous vehicle in vehicular edge computing. *Journal of Cloud Computing*, 10(1):33, jun 2021.

[25] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. Vehicular edge computing and networking: A survey. *Mobile Networks and Applications*, 26(3):1145–1168, Jun 2021.

[26] Lei Liu, Ming Zhao, Miao Yu, Mian Ahmad Jan, Dapeng Lan, and Amirhosein Taherkordi. Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks. *IEEE Transactions on Intelligent Transportation Systems*, 24(2):2169–2182, 2023.

[27] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[28] Pavel Mach and Zdenek Becvar. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, 19(3):1628–1656, 2017.

[29] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358, 2017.

[30] Yuyi Mao, Jun Zhang, S. H. Song, and Khaled B. Letaief. Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. *IEEE Transactions on Wireless Communications*, 16(9):5994–6009, 2017.

[31] MAJA J. MATARIC. Using communication to reduce locality in distributed multiagent learning. *Journal of Experimental & Theoretical Artificial Intelligence*, 10(3):357–369, 1998.

[32] Stephen D. J. McArthur, Euan M. Davidson, Victoria M. Catterson, Aris L. Dimeas, Nikos D. Hatziargyriou, Ferdinanda Ponci, and Toshihisa Funabashi. Multi-agent systems for power engineering applications—part i: Concepts, approaches, and technical challenges. *IEEE Transactions on Power Systems*, 22(4):1743–1752, 2007.

[33] Hamza Ouarnoughi, Emmanuelle Grislin-Le Strugeon, and Smail Niar. Simulating multi-agent-based computation offloading for autonomous cars. *Cluster Computing*, 25(4):2755–2766, Aug 2022.

[34] G.P. Pollini. Trends in handover design. *IEEE Communications Magazine*, 34(3):82–90, 1996.

[35] Zeineb Rejiba, Xavier Masip-Bruin, and Eva Marín-Tordera. Computation task assignment in vehicular fog computing: A learning approach via neighbor advice. In *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, pages 1–5, 2019.

[36] George F. Riley and Thomas R. Henderson. *The ns-3 Network Simulator*, pages 15–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[37] Sylvain Sauvage. Design patterns for multiagent systems design. In Raúl Monroy, Gustavo Arroyo-Figueroa, Luis Enrique Sucar, and Humberto Sossa, editors, *MICAI 2004: Advances in Artificial Intelligence*, pages 352–361, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[38] Philipp Dominic Siedler. The power of communication in a distributed multi-agent system. *CoRR*, abs/2111.15611, 2021.

[39] Andrea Tesei, Marco Luise, Paolo Pagano, and Joaquim Ferreira. Secure multi-access edge computing assisted maneuver control for autonomous vehicles. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–6, 2021.

[40] Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 605–621, Cham, 2020. Springer International Publishing.

[41] Xiaolong Xu, Yuan Xue, Xiang Li, Lianyong Qi, and Shaohua Wan. A computation offloading method for edge computing with vehicle-to-everything. *IEEE Access*, 7:131068–131077, 2019.