Performance Modeling of Vehicular Clouds Under Different Service Strategies

Chinh Tran

A Thesis

In the Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

June 2023

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis is prepared

By: **Chinh Tran**

Entitled: **Performance Modeling of Vehicular Clouds Under Different Service Strategies**

and submitted in partial fulfillment of the requirements for the degree of:

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Chair

Dr. Fuzhan Nasiri

External Examiner

Dr. Jelena Misic

External to Program

Dr. Roch H. Glitho

Examiner

Dr. Dongyu Qiu

Examiner

Dr. Jun Cai

Thesis Supervisor

Dr. Mustafa Mehmet-Ali

Approved by:

Dr. Jun Cai, Graduate Program Director

June 20th, 2023

Dr. Mourad Debbabi, Dean

Gina Cody School of Engineering and Computer Science

Abstract

Performance Modeling of Vehicular Clouds Under Different Service Strategies Using Stochastic Methods

Chinh Tran, Ph. D.

Concordia University, 2023

The amount of data being generated at the edge of the Internet is rapidly rising as a result of the Internet of Things (IoT). Vehicles themselves are contributing enormously to data generation with their advanced sensor systems. This data contains contextual information; it's temporal and needs to be processed in real-time to be of any value. Transferring this data to the cloud is not feasible due to high cost and latency. This has led to the introduction of edge computing for processing of data close to the source. However, edge servers may not have the computing capacity to process all the data. Future vehicles will have significant computing power, which may be underutilized, and they may have a stake in the processing of the data. This led to the introduction of a new computing paradigm called vehicular cloud (VC), which consists of interconnected vehicles that can share resources and communicate with each other. The VCs may process the data by themselves or in cooperation with edge servers.

Performance modeling of VCs is important, as it will help to determine whether it can provide adequate service to users. It will enable determining appropriate service strategies and the type of jobs that may be served by the VC such that Quality of service (QoS) requirements are met. Job completion time and throughput of VCs are important performance metrics. However, performance modeling of VCs is difficult because of the volatility of resources. As vehicles join and leave the VC, available resources vary in time. Performance evaluation results in the literature are lacking, and available results mostly pertain to stationary VCs formed from parked vehicles. This thesis proposes novel stochastic models for the performance evaluation of vehicular cloud systems that take into account resource volatility, composition of jobs from multiple tasks that can execute concurrently under different service strategies. First, we developed a stochastic model to analyze the job completion time in a VC system deployed on a highway with service interruption. Next, we developed a model to analyze the job completion time in a VC system with a service interruption avoidance strategy. This strategy aims to prevent

disruptions in task service by only assigning tasks to vehicles that can complete the tasks' execution before they leave the VC. In addition to analyzing job completion time, we evaluated the computing capacity of VC systems with a service interruption avoidance strategy, determining the number of jobs a VC system can complete during its lifetime. Finally, we studied the computing capacity of a robotaxi fleet, analyzing the average number of tasks that a robotaxi fleet can serve to completion during a cycle. By developing these models, conducting various analyses, and comparing the numerical results of the analyses to extensive Monte Carlo simulation results, we gained insights into job completion time, computing capacity, and overall performance of VC systems deployed in different contexts.

Acknowledgement

Foremost, I would like to express my sincere gratitude to my supervisor, Dr. Mustafa K. Mehmet-Ali, for his invaluable guidance, support, and encouragement throughout my PhD journey. His insights, expertise, and professionalism have been instrumental in shaping my research and my development as a researcher. I am deeply grateful for the time and effort he has dedicated to my work and the trust he has placed in me. I feel fortunate to have had the opportunity to work with him and to have benefited from his mentorship. I would not have been able to complete this thesis without his guidance and support, and I am truly grateful for that.

I would also like to express my sincere gratitude to my thesis committee members: Dr. Dongyu Qiu, Dr. Jun Cai, and Dr. Roch H. Glitho. Their insights, suggestions, and guidance have been invaluable throughout my research. I am deeply grateful for their time and effort in reviewing my work and providing valuable feedback. Their expertise and support have helped shape my research and my research development. I feel fortunate to have worked with such a talented and supportive committee, and I am truly grateful for their contributions to my research.

I also want to express my heartfelt gratitude to Mikhail Ovsyannikov and Duc T. Le for their emotional support during my Ph.D. journey. Their friendship and encouragement have meant so much to me, and have helped me to stay motivated and focused during the challenges of graduate school. I am deeply grateful for their unwavering support and the joy and laughter they have brought into my life. Thank you for being such wonderful friends.

Finally, I would like to express my deepest gratitude to my family members, my mom Hong T. M. Pham, my godmother Ven T. Trần, and my younger brother Vien X. Tran, for their love and support throughout my Ph.D. journey. My mom and godmother have always been great sources of encouragement and shining examples of determination, hard work, understanding, and love. I am grateful for their constant support and for the guidance they have provided throughout my life. I also want to thank my younger brother for being a great source of motivation and for inspiring me to work hard and strive for excellence. I am grateful to have such a loving and supportive family, and I couldn't have completed this journey without them. Thank you, Mom and Vien, for everything.

To my beloved mother, Phạm Thị Minh Hồng

To my dearest godmother, Trần Thị Vén

To my most cherished brother, Trần Xuân Viễn

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

IoT	Internet of Things	MEDs	mobile edge devices
AR	Augmented Reality	PPP	Poisson point process
VR	Virtual Reality	CSMA	carrier sense multiple access
VC	Vehicular Cloud	SDVN	Software Defined Vehicular Networks
VANETs	Vehicular-Ad-hoc-Networks	MHCDD	multi-hop cooperative data dissemination
MCC	Mobile Cloud Computing	VEC	vehicular edge computing
RSUs	Road-side Units	MMPP	Markov Modulated Poisson Process
V2X	Vehicle-to-Everything	PSFFA	pointwise stationary fluid flow approximation
5G	fifth-generation mobile network	PTD	Packet transmission delay
6G	sixth-generation mobile network	PDP	packet delivery probability
SDNs	Software-defined networks	MTTF	mean time to failure
V2V	Vehicle-to-Vehicle	AP	access points
DSRC	Dedicated Short-Range Communications	eCAVs	Electric Connected Autonomous Vehicles
V2I	Vehicle-to-Infrastructure	CPU	central processing unit
eMBB	enhanced Mobile Broadband	UAVs	Unmanned Aerial Vehicles

mMTC	Machine Type Communications	QoS	Quality of service
URLLC	Ultra-Reliable and Low-Latency Communications	CDF	Cumulative Distribution Function
NR-V2X	New Radio Vehicle-to-Everything	i.i.d	independently and Identically distributed
LTE	Long Term Evolution	Pdf	Probability density function
SMC	Secure Multiparty Computation	PGF	Probability Generating Function
MEC	Mobile-Edge Computing	Pmf	Probability mass function
SaaS	Software as a Service		

Chapter 1
Introduction

1.1. Motivation for Vehicular Cloud

The number of smart devices, including vehicles, phones, watches, glasses, and meters, is growing rapidly. New devices are introduced continuously, and they generate massive amounts of data. These devices reside at the edge of the Internet, forming the network known as the Internet of Things (IoT).

Data generated at the edge of the Internet is increasing rapidly due to vehicles equipped with advanced sensors. The generated data is transient by nature, and its significance for potential users decreases rapidly if not processed in a timely manner. The transferring of this data to the cloud is not feasible because of cost and latency [1]. This led to the introduction of the edge computing paradigm to provide computing power close to the source of data. However, this also may not be sufficient as we move towards smart cities [2], which envision providing intelligent services in transportation systems, healthcare, homes, workplaces, and more. Smart cities interconnect city infrastructure, allowing data collection from various human and machine sources. Additionally, recent smart devices like wearables and Augmented Reality/ Virtual Reality (AR/VR) systems, which often execute computationally demanding and time-sensitive applications, further increase the demand for nearby computing resources. The modern vehicles are having high computational power and storage capabilities. These resources may be under utilized which may be used to address the rising computational needs at the edge of the Internet [3], [4]. The cluster of computation and interconnectivity based on a network of modern vehicles is called a vehicular cloud (VC). As a result, the emerging vehicular cloud concept offers a promising solution, combining the untapped computational resources of the vehicles to tackle the challenges related to the generation of large amounts of data and latency requirements [5]–[7].

Vehicular cloud technology goes beyond simply offering surplus computational resources to nearby devices; it also presents potential solutions to pressing societal issues like traffic congestion and accidents. In the following sub-sections, we will explore the various applications

of vehicular cloud that can lead to safer and more efficient transportation systems as well as creative value-added services and applications.

1.1.1. Improved traffic management

The exponential rise in vehicular traffic in recent years has placed considerable strain on existing traffic management systems [8], prompting the search for innovative solutions to alleviate congestion and improve overall traffic conditions.

It has been demonstrated that incorporating vehicular cloud technology into traffic management systems can substantially improve traffic flow and reduce congestion. This is largely due to the ability of vehicular cloud networks to collect and share real-time traffic information from numerous vehicles, which allows traffic management systems to make more informed decisions [9].

1.1.2. Enhanced road safety

Vehicular cloud may help to enhance road safety and reduce traffic accidents [10], [11]. One of the primary motivations for research in this domain is the rising number of fatalities due to traffic accidents, which has been a global concern for years [12]. Vehicular cloud systems will facilitate the exchange of critical data related to traffic conditions and road hazards, which can be instrumental in developing predictive models and early warning mechanisms to mitigate accidents. By pooling the sensing capabilities of multiple vehicles, the vehicular cloud can create a more comprehensive and accurate picture of the road environment. Group sensing can help identify potential risks and hazards, such as road obstructions or pedestrians, leading to improved safety measures [13]. In the event of an accident, the vehicular cloud can streamline communication between emergency services and connected vehicles, ensuring a faster and more efficient response. Streamline communication can help minimize the severity of accidents and improve overall road safety [14], [15]. As such, exploring the potential of the vehicular cloud for improving road safety is essential for supporting safer and more efficient transportation ecosystems.

1.1.3. Enabling new services and applications

One of the most compelling aspects of vehicular cloud computing is its ability to facilitate dynamic and adaptive resource sharing, which can revolutionize providing services such as in-vehicle infotainment and cooperative sensing [16], [17]. Additionally, the vehicular cloud enables the proliferation of location-aware services, thus enhancing the overall driving experience by providing real-time information about traffic conditions, parking availability, and locations of interest [18]. Furthermore, vehicular cloud computing allows for seamless, ubiquitous connectivity. This enables the development of many location-based services, significantly improving the overall driving experience [19]. However, extensive research in various aspects of the field is still necessary to realize these value-added applications. For example, it is essential to explore advanced techniques for efficient management of vehicular resources, as in [20], [21].

1.2. Architectures of Vehicular Cloud

Vehicular Cloud architecture has evolved since its conception due to the advancement in vehicular design and technology, communication capabilities, and potential applications of VC. Thus, it encompasses many different architectures, such as Vehicular Cloud Computing, Vehicular Fog Computing, and Vehicular Edge Computing. Vehicular Cloud Computing is originally an evolution of Vehicular-Ad-hoc-Networks (VANETs) inspired by Mobile Cloud Computing (MCC) [22], which allows mobile devices, such as mobile phones and laptops, to leverage cloud computing services and resources [23]. Eventually, as the number of data-generating devices surged unexpectedly, Cloud Computing struggled to process these data in real-time due to network bandwidth constraints. Consequently, Edge Computing [24] and Fog Computing [25] emerged almost simultaneously to address the challenge in two different manners. While Edge Computing proposes a pervasive and miniaturized version of Cloud Computing and still permits users to leverage the processing of external edge servers nearby, Fog Computing proposes a network of user devices to interconnect and "co-operate among them and with the network to perform storage and processing tasks without the intervention of third-parties." Vehicular Edge Computing and Vehicular Fog Computing are the subsets of Edge Computing and Fog Computing, respectively, where the users are the vehicles. However,

specific to the vehicular domain, edge servers can be placed at Road-side Units (RSUs) to serve moving vehicles [26].

VC can also be classified into two categories called static or stationary VCs and dynamic VCs. Static VCs are usually based on stationary vehicles in parking lots, while dynamic VCs are based on moving vehicles on highways and roads in the cities.

As there is yet a realization of vehicular cloud, a VC architecture that is more recent and evolved does not necessarily replace the older architectures. Thus, this section intends to investigate the details of three primary vehicular cloud architectures: centralized, distributed, and hybrid. Our goal is to dissect the unique attributes of each architectural type while assessing their respective strengths and weaknesses concerning scalability, reliability, and communication efficacy. By examining these architectures in depth, we aim to provide a comprehensive understanding of the design principles supporting the development of robust and efficient vehicular cloud systems.

1.2.1. Centralized architecture (Vehicular Edge Computing)

In the realm of vehicular cloud computing, the centralized architecture is characterized by a central server, which is often a nearby edge computing system that serves the communication and computation needs of vehicles and other network devices [27], [28].

The primary strength of the centralized architecture is its potential for improved communication efficacy, as the single control entity can efficiently coordinate data transfer and computation among vehicles [29]. Moreover, the centralized nature of this architecture facilitates better resource allocation and task prioritization, which can lead to more efficient resource utilization and overall performance improvements [20]. Another advantage is the ease of implementing security measures, as the central control entity can oversee and enforce security protocols across the entire network, ensuring a consistent and robust security framework.

Nevertheless, this architecture also exhibits certain drawbacks, particularly with regard to scalability and reliability. As the vehicular cloud expands and the number of connected vehicles grows, the central control unit may struggle to accommodate the increased computational workload, leading to possible congestion and performance issues. Additionally, the centralized

nature of this architecture makes it vulnerable to single-point failures, as any disruption in the central control unit can have significant consequences for the entire network.

In conclusion, the centralized architecture of the vehicular cloud offers benefits in terms of communication efficacy, resource allocation, task prioritization, and security but may face significant challenges regarding scalability and reliability.

1.2.2. Distributed architecture (Vehicular Fog Computing)

In the modern era of vehicular technology, the distributed architecture of the vehicular cloud has emerged as a promising solution to harness the potential of computing resources available in vehicles [7]. The distributed architecture of the vehicular cloud is based on the assumption that vehicles are capable of handling offloaded computation, albeit with opportunistic resources [30]–[37]. This architecture is characterized by its ability to utilize the computing power of individual vehicles despite their resource intermittence nature.

Due to the distributed characteristics of the architecture, the system does not suffer detrimentally when any single vehicle or component fails. In other words, this system's fault tolerance capability is higher than the centralized system, which enables continuous operation under this architecture. Additionally, since the computational workload is shared among multiple vehicles, this system makes better use of under-utilized resources on vehicles [38]. Moreover, the distributed architecture facilitates data processing closer to the source. This reduces energy consumption from data transmission. Furthermore, a distributed architecture allows the system to scale easily. A vehicle with computational resources and communication capabilities can easily and seamlessly be integrated into the computing network. This flexibility to incorporate new computing resources also means that utilization of the resources is opportunistic, and their availability is time-varying. The increased reliability comes at the expense of increased complexity in terms of communication and coordination among the vehicles, which could potentially impact the overall efficacy of the communication process.

On the other hand, the pool of resources from the vehicles can vary dynamically due to various reasons, such as congested communication channels or intermittent availability of computing resources, which may penalize the overall performance of the vehicular cloud. As the processing load is distributed across multiple vehicles, delays in communication between these

vehicles could negatively impact the speed at which tasks are completed. Moreover, the varying capabilities of individual vehicles in terms of computing power and resource availability may lead to an uneven workload distribution, further intensifying the latency issue [5].

1.2.3. Hybrid architecture

The hybrid architecture of the vehicular cloud allows offloading of the computational tasks among vehicles, edge servers, and cloud servers, enabling dynamic allocation of resources based on the workload at any given time [39]–[42]. By leveraging the unique capabilities of these different entities, hybrid architecture aims to create a highly efficient and adaptive computing environment to meet the ever-increasing and diverse demand for computation power by new applications.

The architecture shares multiple advantages with the distributed architecture, such as scalability, fault-tolerant, resource management, and scheduling flexibility. In this system, each component, including the vehicles, edge servers, and cloud servers, can autonomously contribute to the computational demands of the network, thereby mitigating the potential for bottlenecks in processing capabilities [43]. Furthermore, it is almost always possible to find more resource options for a specific task due to the diverse resource pool [44]. Moreover, hybrid architecture is quite reliable because of its decentralized design. Single points of failure are distributed across different computing platforms, creating a more stable system that is less likely to experience major failures.

Hybrid architecture is not without its limitations. The communication efficacy between vehicles, edge servers, and cloud servers may be hampered by latency issues, particularly in scenarios where rapid real-time data processing is essential for optimal functionality. Another potential issue with this architecture is the complex orchestration required to synchronize and coordinate the dynamic offloading of computational tasks among vehicles, edge servers, and cloud servers, thereby causing operational inefficiencies and worsening resource allocation challenges. Another concern is the increased risk of security vulnerabilities in the system. The larger and more diverse a system becomes, the more interfaces and vulnerabilities it might have. Thus, there will be many more potential attack vectors to compromise the system.

1.3. Enabling technologies of Vehicular Cloud

The vehicular cloud cannot emerge without a foundation laid by other enabling technologies since the vehicles have not always been computationally capable, and the communication channels have not been advanced enough to support tasks offloading. Recently, wireless communications, such as fifth-generation mobile network (5G) and sixth-generation mobile network (6G) technologies and vehicle-to-everything (V2X) communication, have played a pivotal role in facilitating the seamless exchange of information between vehicles and their surrounding environment. The computation capacities of the vehicles have increased drastically to support vehicular autonomy. Furthermore, advanced vehicular sensor systems have emerged as an enabling technology, encouraging data from the sensor system to be processed by nearby vehicles, which enhances situational awareness and cooperative decision-making in the vehicular cloud. Software-defined networks (SDNs) offer adaptable resource management, enhancing control and resource sharing in vehicular cloud operations.

Next, we will describe each of the enabling technologies mentioned above that is contributing to the development of vehicular clouds.

1.3.1. Advanced vehicular sensor systems

In the present era, advanced sensor systems have emerged as indispensable components of modern vehicles, contributing significantly to enhanced safety, performance, and efficiency. Among the collection of sensors integrated into modern automobiles, radar, lidar, camera, and ultrasonic sensors feature prominently, working together to achieve optimal vehicular operation [45]. At the same time, the sensors produce an extremely large amount of data, which is incredibly rich in contextual information, extremely valuable, highly time-sensitive, and location bound. As such, the data must be processed immediately after its generation. However, the colossal amount of data generated by these sensor systems poses a daunting challenge, as the movement of this amount of data to the cloud is not feasible due to cost and bandwidth limitations.

1.3.2. Wireless communication

The evolution of wireless communication enabling vehicular cloud systems started with VANETs. Initially, VANETs were dependent on Vehicle-to-Vehicle (V2V) communication, facilitated by Dedicated Short-Range Communications (DSRC). As the standards facilitating VANETs evolved, which allowed vehicles to interact with surrounding infrastructure improved, Vehicle-to-Infrastructure (V2I) paradigm emerged.

The introduction of 5G, the fifth-generation mobile network, played a significant role in the evolution of vehicular communication. 5G offers superior speed, latency, and reliability compared to its predecessors and is based on three pillars: enhanced Mobile Broadband (eMBB), massive Machine Type Communications (mMTC), and Ultra-Reliable and Low-Latency Communications (URLLC). The development of 5G has been crucial in fostering V2X communication [46], [47], which aims to create a seamless flow of information among the various elements in the transportation ecosystem.

V2X communication encompasses the exchange of information between a vehicle and any other entity, including other vehicles, infrastructures, and people. V2X enables vehicles to communicate with smart objects that belong to IoT. URLLC, the building block of V2X communication, has significantly enhanced the capabilities and potential of vehicular cloud systems by enabling real-time data sharing among vehicles. Recent studies have demonstrated the superiority of New Radio Vehicle-to-Everything (NR-V2X) over Wi-Fi-based technologies, such as DSRC, in terms of data rate and latency [48]. The study shows that NR-V2X can achieve 10 ms latency for 100 bytes and 1500 bytes packets for a communication range of up to 500 m. Additionally, the study also shows that NR-V2X can achieve around 2 Mbps transmission rate for non line of sight (NLOS) for 500 m, which is an order of magnitude difference compare to Wi-Fi-based technologies. Other research has highlighted the advantages of Long Term Evolution (LTE) - based V2X communication, including wide coverage, high capacity, and high penetration [49]. More recently, empirical results show that LTE-V2X outperforms DSRC at long range and higher traffic density [50], which enables further the realization of VC.

5G's key contribution to vehicular cloud systems is the enablement of V2X communication, fostering connectivity and collaboration among vehicles, infrastructure elements, and other road users. The ultra-low latency characteristics of 5G networks support

time-critical applications like collision avoidance, emergency response, and intelligent traffic management [46]. Additionally, the high data rates offered by 5G networks significantly enhance the computing capabilities of vehicular cloud systems, enabling the sharing of multimedia content, advanced mapping information, and real-time data analytics [47]. By seamlessly integrating 5G and V2X communication technologies, vehicular cloud systems can provide vehicles with faster access to cloud computing resources, thereby improving the efficiency of onboard computing units [51].

The significance of V2X in overcoming the challenges posed by unreliable communication channels in the Vehicular Cloud cannot be overstated. By seamlessly connecting vehicles and enabling resource sharing, V2X serves as the critical part of establishing a robust and dependable distributed network of computing power, storage, and supplementary services like location and witnesses [52]. V2X communication can also facilitate traffic flow optimization, enhances situational awareness, and enables cooperative driving applications, such as platooning or smart intersections [53]. This facilitation would reinforce and further stabilize the communication channels in Vehicular Cloud. Additionally, by providing vehicles with access to localized services and content, V2X technology significantly reduces dependence on continuous Internet connectivity, leading to a more stable and efficient network [54].

1.3.3. Autonomous vehicles

Autonomous vehicles, commonly referred to as self-driving cars, represent a paradigm shift in transportation, leading the way to a new era of mobility. These sophisticated machines employ a wide range of sensors, algorithms, and advanced technologies to make their way through city streets and highways and make split-second decisions autonomously without relying on human input [4]. Consequently, autonomous vehicles promise to considerably mitigate traffic-related accidents, enhance fuel efficiency, and improve overall traffic conditions [55] while also potentially playing an enormous part in enabling vehicular cloud.

For example, autonomous vehicles can exchange data to maintain optimal distances, circumvent collisions, and reduce redundant sensory resources. The latter perk permits the reduction in overall computation utilization rate if the vehicles are not connected, allowing excess computing to be leveraged by external devices. Furthermore, they can collaborate in

orchestrating traffic flow, thereby mitigating congestion and minimizing emissions. Additionally, the immense processing power of autonomous vehicles can be harnessed to examine and combine massive amounts of data, thereby facilitating the emergence of intelligent urban ecosystems and cutting-edge transportation infrastructures. In sum, the emergence of autonomous vehicles stands not only to revolutionize transportation and urban living in myriad ways but also to support the emergence of vehicular clouds.

1.3.4. Software-defined networks (SDNs)

SDNs paved the way for a transformative change in network management and orchestration, offering a more dynamic and programmable approach to controlling network resources [56]. By decoupling the control plane from the data plane, SDNs promote a centralized system for configuring, managing, and optimizing network infrastructure. The myriad advantages of SDNs, encompassing flexibility, scalability, and resource optimization, have brought about a wide array of applications spanning cloud computing, the IoT, and vehicular networks, including vehicular cloud

Software-defined networks play an instrumental role in facilitating vehicular cloud deployment. Firstly, SDNs foster a more robust and efficient connection between vehicles, thereby augmenting the overall vehicular cloud performance [57]. This is achieved through the optimization of routing paths and the utilization of real-time information for adaptive network infrastructure adjustments. SDNs are capable of managing and directing traffic in a distributed manner, thus improving network resilience and flexibility. Further, SDNs also bolster the security of the vehicular cloud by enabling distributed threat detection and mitigation [58]. Network traffic can be analyzed in various parts of the network for abnormal patterns, allowing for a more comprehensive and efficient response to potential threats. Thirdly, SDNs facilitate the management and allocation of vehicular cloud resources. By centralizing control, administrators can effortlessly monitor the network's available resources of the vehicular clouds and allocate them based on demand, mitigating resource contention and augmenting overall performance [59]. Lastly, SDNs contribute to the scalability of the vehicular cloud. As the number of vehicles surges, the network configuration can be easily adjusted and expanded to cater to growing requirements [60]. In summation, software-defined networks are indispensable in actualizing

vehicular cloud and possess the potential to revolutionize vehicular communication and resource-sharing paradigms.

1.4. Security and privacy in Vehicular Cloud

Vehicular cloud computing demands a thorough focus on security and privacy aspects. In fact, security and privacy in vehicular clouds directly impact the passengers onboard, making them two major obstacles in the implementation of vehicular cloud technology. Thus, ensuring data security and integrity is a primary concern, requiring strong encryption techniques and robust communication infrastructures to protect vehicular data against malicious actors and breaches. Additionally, adopting privacy-protection methods, including data anonymization, obfuscation, and aggregation techniques, is crucial for user anonymity and the reduction of personal information exploitation. The brief literature review below outlines possible strategies to tackle security and privacy issues in vehicular cloud, thereby increasing the feasibility of this paradigm.

Data security and integrity are vital in information technology. Implementing strong encryption algorithms is essential for secure vehicle communication, preventing eavesdropping, tampering, or illegal data access [61]. Deployment of secure authentication protocols is necessary for the confirmation of vehicle identities and the prevention of unauthorized access [62]. Data integrity involves guaranteeing unaltered and consistent information, using techniques like checksums, digital signatures, and error-correcting codes [63]. Preserving data integrity is crucial for smooth vehicular cloud operation, as network-provided services rely on exchanged data accuracy [64].

Privacy-preserving techniques are essential in modern digital environments. In vehicular cloud, these techniques ensure data privacy for exchanged data volumes, including geolocation, driving patterns, and vehicle diagnostics. Encryption can secure communication channels, ensuring access solely for authorized entities [65]. Differential privacy can obscure individual data points within aggregate datasets, preserving user privacy while providing valuable insights [66]. Privacy-preserving techniques enable secure and private computation, using methods like secure multiparty computation (SMC) and homomorphic encryption for computations on encrypted data, preserving sensitive data confidentiality even during processing [67]. These

methods promote trust between users and service providers, driving widespread vehicular cloud technology adoption.

1.5. Resource Management in Vehicular Cloud

Effective resource management plays an important role in vehicular cloud computing, as it directly impacts the performance, efficiency, and quality of services offered by this technology. More importantly, by employing well-designed resource management methods, vehicles can be given more incentives to share their resources, promoting greater collaboration and resource pooling within the network. The motivation for resource management in vehicular cloud is to maximize system performance, minimize latency, and improve overall reliability while addressing the challenges of limited availability, dynamic topology, and diverse application requirements. In this context, various resource allocation and scheduling techniques have been proposed to achieve these objectives, which can be categorized into six groups: game theoretic approaches, auction-based mechanisms, learning-oriented optimization techniques, optimization-based strategies, bio-inspired algorithms, and consensus-based methodologies.

Game theory provides a mathematical instrument for interpreting and examining strategic interactions amongst multiple rational actors within a system [68]. Concerning vehicular clouds, game-theoretic principles have been employed to investigate cooperative and adversarial engagements between vehicles to attain optimal resource allocation [69]. Importantly, game theory provides incentives for vehicles to share their resources, promoting more efficient utilization and distribution of vehicular resources. Various game-theoretic frameworks have been applied to vehicular cloud resource allocation, including cooperative games, non-cooperative games [70], and coalitional games, each with its distinctive characteristics and assumptions. Additionally, solution concepts, such as Nash equilibrium [71] and Pareto efficiency, have been employed to analyze the outcomes of strategic interactions among vehicles, thus facilitating stable and efficient resource allocation schemes.

Conversely, auction mechanisms offer a decentralized methodology for resource allocation [72], [73]. Under this schema, vehicles can function as buyers and bidders, tendering bids for accessible resources to satisfy their computational requisites. Auction-based mechanisms provide incentives for vehicles to share their resources, as vehicles can benefit from pooling

resources together to increase their bidding power and maximize their chances of acquiring the required resources. Auction-oriented systems can foster efficient resource allocation by integrating market forces and stimulating competition amongst vehicles. Various auction formats, such as double auctions [72], [74], have been proposed to facilitate resource allocation in vehicular cloud environments. These auction mechanisms can be further refined through the incorporation of incentive-compatible mechanisms, pricing schemes [75], and truthful bidding strategies [76], ultimately prompting a fair and transparent resource allocation process.

Learning-centric optimization techniques are a promising approach to resource allocation in vehicular clouds [27], [46], [77]–[79]. Such methods employ machine learning algorithms to extrapolate knowledge from past experiences and adapt resource allocation strategies accordingly. Incorporating learning-based optimization enables vehicular cloud systems to accomplish superior resource allocation outcomes in dynamic and uncertain environments. Various learning techniques, such as reinforcement learning [32], [46], [77], have been employed to enhance resource allocation decisions in vehicular clouds. These algorithms can leverage historical data and feedback mechanisms to continuously refine and update resource allocation policies, ensuring a robust and adaptive response to changing vehicular network conditions.

Optimization-based strategies are grounded in mathematical models designed to maximize or minimize an objective function while adhering to specified constraints. More specifically, they intend to efficiently allocate computing, storage, and communication resources among connected vehicles in a vehicular cloud network to solve computationally intensive tasks [44]. These methods aim to maximize the overall performance and minimize the latency while taking into account the constraints and requirements specific to vehicular environments, such as high mobility, limited resources, and variable network conditions [34]–[36]. By leveraging mathematical optimization techniques, such as linear programming [33], [80], integer programming [81], and dynamic programming [82], researchers can attain optimal resource allocation solutions for vehicular cloud systems. Moreover, multi-objective optimization approaches, which consider multiple conflicting objectives simultaneously, have been proposed to account for the trade-offs inherent in vehicular cloud resource allocation, ensuring a comprehensive process [83].

Bio-inspired algorithms, as the name suggests, are heuristics and metaheuristics derived from natural phenomena, emulating the processes observed in biological systems. These algorithms, which include genetic algorithms [20], [37], [84], swarm intelligence [85], [86], and ant colony optimization [87], have been used in vehicular cloud resource allocation due to their adaptability and ability to find near-optimal solutions. These algorithms help manage the complexity of resource allocation in vehicular cloud systems. Hybrid bio-inspired algorithms, which combine multiple techniques, improve performance and efficiency. Examples include combining genetic algorithms with particle swarm optimization or ant colony optimization, enabling more efficient exploration and exploitation of the solution space [88]. As a result, bio-inspired algorithms are well-suited for addressing the challenges of vehicular cloud resource allocation.

Consensus-based methodologies employ distributed coordination and decision-making principles to achieve consensus on tasks allocation amongst vehicles [39], [89]. Vehicles in a network communicate, share information, and iteratively update their local knowledge to converge on a collective agreement for tasks allocation. This approach to resource allocation fosters a self-organized, decentralized system in which vehicles can collaborate and synchronize their actions to attain optimal tasks distribution. Various consensus algorithms, such as the average consensus algorithm [90], the weighted consensus algorithm [91], and the gossip-based consensus algorithm [92], have been utilized to facilitate resource allocation in vehicular cloud systems. Additionally, fault-tolerance mechanisms and privacy-preserving techniques can improve the robustness and security of consensus-based resource allocation, ensuring reliable and trustworthy resource distribution in vehicular networks.

1.6. Performance Modeling of Vehicular Cloud

As explained in the previous section, resource management deals with the allocation of computational resources to tasks such that the system performance is maximized. In general, resource management assumes that available resources and task workloads are constant. Given the available resources and workload, it attempts to determine the optimal allocation of the resources to the tasks. Thus, random processes that govern the system's behavior, such as the

arrival process of vehicles to the VC, residency times of the vehicles, number of tasks in a job, and task execution times, are not captured.

Performance modeling of vehicular cloud systems attempts to evaluate system performance by taking into account its stochastic nature. Job completion time, resource utilization, and service capacity of vehicular cloud are important performance measures. There are three primary methods of performance modeling: stochastic modeling, simulation, and statistical analysis. Stochastic modeling offers a mathematical framework to study the performance of vehicular clouds [93]–[97]. Simulation, on the other hand, creates a digital environment replicating real-world vehicular cloud scenarios, enabling researchers to analyze various configurations and identify potential bottlenecks without implementing them in an actual system [39], [44], [98]. Lastly, statistical analysis involves examining data from real-world vehicular cloud systems to identify trends and patterns in traffic, which can inform the design of increasingly more efficient architectures [6].

As the main subject of this thesis is performance modeling using the stochastic modeling method, detailed reviews of relevant works on the same subject matter will be discussed further below in two different subsections divided by vehicular cloud architectures.

1.6.1. Hybrid or Centralized VC architectures

In [99], the authors address the growing demand for powerful mobile applications, such as VR and AR, which are often limited by storage, computing capability, and battery life of mobile devices. The paper focuses on an application scenario where Mobile-Edge Computing (MEC) facilities are provided as a Software as a Service (SaaS) within a limited geographical area for mobile-edge devices (MEDs). The authors employ a Markov model with requests reneging to analyze the MEC system performance concerning task-dropping probability and mean time spent by a computation request in the MEC system. They develop a design procedure to identify the optimal number of computation resources to be allocated to the MEC system to fulfill specific QoS requirements. However, the analytical model relies on several simplifying assumptions, making the problem more manageable but potentially limiting the model's applicability in more complex real-world scenarios.

[100] investigates the local delay problem in a MEC-based VANET, specifically focusing on a suburban scenario where a vehicle node requires computing services from an edge node.

The authors derive closed-form expressions for both uplink and downlink local delays using stochastic geometry. Key assumptions include the spatial distribution of vehicle nodes on each street following an independent 1-D homogeneous Poisson point process (PPP) and edge nodes being deployed at each intersection with a certain probability. Contrary to most existing local delay models for VANETs, the authors employ a carrier sense multiple access (CSMA) mechanism for channel access by both vehicle and edge nodes. The derived analytical model's effectiveness is verified through numerical results, which also investigate the impacts of underlying network parameters on the local delay. The model serves as a guideline for edge node deployment. Despite its contributions, the work focuses exclusively on a suburban scenario, leaving the analysis of an urban scenario to be explored in future research.

The work presented in [101] revolves around the performance modeling of a single-server MEC system under a task scheduling strategy that considers both the priority and time constraint of computing tasks. The authors construct a 3-D Markov chain to depict the queuing and processing process at a MEC server, accommodating the task scheduling strategy. From this Markov model, they derive the system's performance in terms of the average drop probability and average waiting time. However, the work has its limitations, as it focuses on a single-server MEC system, which may not be representative of more complex MEC systems.

In [102], the authors propose the use of Software Defined Vehicular Networks (SDVN) to address the challenges posed by the significant increase in traffic within VANET, which has resulted in considerable delays in data transfer. By employing reliable multi-hop cooperative data dissemination (MHCDD) for data traffic forwarding, SD-VANETs, a form of SDVN, enhance end-to-end connectivity and network resource utilization in dynamic network topologies. The authors analyze the network model using $M/M/m$ queueing model to effectively minimize delay and enhance throughput within the multi-server queueing system of SDVN.

[103] presents an analytical model for evaluating the performance of vehicular edge computing (VEC) systems with bursty task arrivals. It's assumed that the tasks may be executed locally, at neighboring vehicles and edge servers. It's assumed that the vehicles generate tasks according to Markov Modulated Poisson Process (MMPP). Each vehicle keeps locally generated tasks and the offloaded tasks from the neighboring vehicles in separate queues. They use a Markov chain analysis to determine the average task delay. The analysis in this work does not

consider jobs with multiple tasks, and there is no resource volatility as the number of vehicles is assumed to be constant.

In [104], a cooperative computation model is studied between the nearby vehicles and edge servers. They consider three node computation model where the nodes are two vehicles and an edge server. One of the vehicles is the host vehicle for the task, and the other vehicle and the edge server are within one-hop communication range of the host vehicle. The authors consider three execution modes of the task: by a single node, by two nodes, and by all three nodes. In the cases of multiple node executions, the task is partitioned into subtasks, each to be executed by a node. The analysis also takes into account communication reliability and overhead. They determine the probability of successful completion of a task through constrained optimization. Their results show that jobs with short deadlines and large data sizes will benefit from cooperative computation as opposed to noncooperative solutions.

1.6.2. Distributed VC architecture

In [105], the authors analyze a VC based on the vehicles at a traffic-light-controlled intersection. The authors examine an intersection's traffic by modeling it as a G/G/1 queue. Then, they analyze it with the diffusion method. Based on this traffic model, the distribution of the lifetime of the VC that follows a specific quality of service criteria, which is that number of vehicles must be above a certain threshold, is determined. However, the task service strategy and the performance of the VC under computational load have not been studied.

[106] presents a model to evaluate the time-dependent performance of platooning communications at intersections using the 802.11p protocol in autonomous driving scenarios. The authors consider various movement behaviors of platoons at intersections, including turning, accelerating, decelerating, and stopping, as well as the periodic change of traffic lights. Subsequently, they establish a hearing network to reflect the time-varying connectivity among vehicles. The pointwise stationary fluid flow approximation (PSFFA) is employed to model the nonstationary behavior of the packet transmission queue. The packet transmission delay (PTD) and packet delivery ratio (PDR) are derived, and the accuracy of the proposed model is validated through simulation.

In the [107], the authors examined the performance of broadcast packets in a one-dimensional VANET using the slotted ALOHA protocol. The analysis primarily centered on the packet delivery probability (PDP) under a delay constraint, given its importance in safety-related applications within VANETs. The authors used stochastic geometry theory to represent vehicle locations and introduced an approximation technique to address spatial correlations in interference. The authors showcased, via numerical and simulation findings, that spatial correlations in interference have a notable impact on PDP and produced an analytical formula for PDP that can be numerically computed.

In [108], [109], the authors investigated job completion time in a stationary VC in a parking lot. It is assumed that the execution time of a job is a random variable with a finite mean, and the residency time of the vehicles in the VC is exponentially distributed. For reliability, the system uses redundancy which assigns a job to two vehicles. Each of the vehicles executes its instance of the job. When one of the vehicles departs before finishing the execution of the job, the remaining vehicle halts its execution, and the work done by the leaving vehicle is saved. Then, the system starts recruiting an additional vehicle to resume the execution of the job. The recruitment duration is a random variable with a general distribution. If a new vehicle is recruited before the remaining vehicle departs, then the saved work is migrated to the new vehicle. Following that, both vehicles resume the execution of the job. This iteration of the execution fails if the remaining vehicle leaves before another vehicle is recruited. The unfinished work is discarded, and a new iteration begins with the recruitment of two new vehicles for the execution of the job. This process continues until the execution of the job is successfully completed. The paper determines the expected completion time of job execution.

The work in [110] is a follow-up of [108] and it uses the same assumptions. They propose an alternative approach for the derivation of job completion time that takes into account the cost of failures. This is done by assuming that approximations to the first and second-order moments of the parameters are available. They consider three scenarios: both the first and second moments of both the duration of the recruitment and execution time of a job are available; only the first moment of the job execution time is known; the first and second moments of the job execution time are known. The scenario that assumes the availability of the first and second moments of job execution time seems to follow the simulation results better than the other

scenarios. However, for longer job execution times, the correlation between the approximations and simulations is not good.

In [111], the availability aspect of a stationary VC in a parking lot is analyzed using stochastic modeling. Jobs are serviced under "Jn" strategy, where each of the n copies of a job is executed by different vehicles. The work determines a closed-form expression for the mean time to failure (MTTF) of Jn job assignment and presents a formulation to determine n given a QoS threshold.

Though the works in [108], [110], [111] derive job completion time in a VC, the models assume a stationary VC, and the job consists of a single task. The assumption of simultaneous execution of a job by multiple vehicles increases job completion time substantially. Since the model assumes a stationary VC with parked vehicles, this redundancy seems to be unnecessary, as communication in this environment should be reliable.

In [112], the authors investigate job completion time for a dynamic VC system on a highway with a service without redundant execution and migration overhead. It is assumed that there are access points (AP) on the highway with arbitrary coverage, but there will be segments of the highway with no coverage. The vehicles can only upload and download their work under the coverage of an AP. The authors also assume that the downloading and uploading can only occur under the coverage of a single AP, and a job can be completed by a single vehicle without communicating with other vehicles. The authors list three scenarios where a vehicle completes its job: a vehicle finishes a job under the coverage of one AP but finishes the uploading of the result under the coverage of the next AP; job execution ends outside of the coverage of an AP; job execution and uploading of the result is completed under the same AP. Then, the authors derive the completion time of a job in each case. After that, they combine the results of the three cases by weighing them with their limiting probabilities. The authors then confirm their findings by simulation. Though this work studies job completion time in a dynamic VC, they assume that the residency time of the vehicle does not expire during the execution of a job and a job consists of a single task.

In [113], they study energy-aware resource management in Electric Connected Autonomous Vehicles (eCAVs). It's stated that these vehicles will generate a massive amount of data that needs to be processed with minimum latency. This problem may be addressed by

having edge servers located at RSUs. However, this solution may not be scalable due to a large number of vehicles, workload, and limited capacity of edge servers. Local execution of the workload by operating a central processing unit (CPU) at a high frequency, even if it meets the requirements, is not desirable because it will result in high power consumption, which will reduce the driving range of the electric vehicle. It has been found that a task executed at a high CPU frequency may consume much more power than executing a task at a lower CPU frequency. As a result, energy savings may be achieved by partitioning the tasks into subtasks and executing them in multiple vehicles in parallel at lower CPU frequencies. The authors propose an energy management algorithm that minimizes computational energy by distributing the workload among a number of eCAVs.

1.7. Research objectives

In this section, we explain the objectives of the research in this thesis. The widespread realization and acceptance of VCs still face many challenges. These challenges arise due to the volatility of resources, privacy concerns, and drivers not willing to share their resources. However, the realization of many societal goals requires the assistance that VCs may offer. Resource volatility arises from intermittent communication connectivity and random residence times of the vehicles in the VCs. Advances in wireless communications, installation of RSUs, and deployment of Unmanned Aerial Vehicles (UAVs) [114] will help maintain connectivity in VCs. Smart vehicles and smart cities will benefit from cooperative driving in data collection and forming a view of the environment. These will provide incentives for drivers to share their resources and remain members of a VC for a longer period of time. Thus, drivers may be willing to form a platoon by adjusting their speeds to prevent the breakup of the VCs. In the future, VCs will be more stable and have longer lifetimes.

The objective of this thesis is the performance evaluation of these networks under different operating environments and workloads. Important performance measures of VCs are job completion time and their computation capacity. Knowledge of job completion time will help inform deployment strategies and ensure that QoS requirements are met. The computation capacity of a VC indicates the number of jobs that can be completed during the lifetime of this system. The complexity of vehicular cloud systems presents a significant challenge to their

performance modeling. Next, we discuss the factors that impact the performance of these systems.

- **Ressource volatility**: As stated in the above, the behavior of these systems can vary greatly depending on the specific workload and environment in which they operate. Since vehicles are moving in and out of the vehicular cloud randomly in these systems, the available resources are time-varying. The variability of the resources depends on whether VCs are dynamic or static. The volatility of the resources will be higher in dynamic VCs than in static ones. Even in dynamic VCs, resource volatility depends on whether VCs are formed on highways or on streets with traffic lights and stop signs. Also, whether the traffic is free-flowing or congested impacts the availability of resources. The residency time of vehicles in congested traffic will be longer. The drivers caught in congested traffic will be more willing to share their resources, especially if the objective of an application is to alleviate the congestion.
- **Job Composition**: The performance of the system depends on whether a job consists of single or multiple tasks. In the case of a job with multiple tasks, whether these tasks are independent of each other or dependent on each other also impacts the performance. If the tasks of a job are independent, they can be executed concurrently without waiting for the results from the other tasks.
- **Service strategy**: Another important factor that impacts the performance of the system is the task service strategy. In this case, we have three service strategies: interrupted, uninterrupted service, or hybrid task service. In the interrupted service strategy, a task is assigned to a vehicle even if the residency time of the vehicle is not sufficient to finish the execution of the task. In this strategy, when the vehicle departs, the task is assigned to another vehicle, which resumes its execution from where it was left. This strategy suffers from migration overhead. In the uninterrupted service strategy, a task is assigned to a vehicle only if the residency time of the vehicle is sufficient to finish the execution of the task. Thus, this strategy does not experience migration overhead due to service interruption. However, if the execution time of a task is long compared to vehicle residency times, then the task assignment may take a long time. The job completion time increases due to migration overhead in the interrupted service strategy and due to task assignment delay in the uninterrupted

service strategy. A hybrid service strategy is a mix of interrupted and uninterrupted service strategies. It uses an interrupted service strategy to assign tasks with long execution times and an uninterrupted service strategy to assign tasks with short execution times. An important parameter will be the threshold that determines which service strategy to use.

From the literature survey in the previous section, research work on the performance evaluation of VCs is very limited. The references [108]–[111] determine the average job completion time in a stationary VC in a parking lot. They assume a service strategy with interruption, thus, a vehicle may depart before execution is completed. The drawback of this work is that there is no vehicle arrival process to the VC. It is assumed that the amount of time to recruit a vehicle to execute the job is a random variable with a general distribution. The recruitment time is assumed to be the same whether or not there are vehicles in the parking lot. Another drawback of these works is the assumption that the job consists of a single task.

The work in [112] determines job completion time in a dynamic VC on a highway scenario. It is assumed that there are APs along the highway; however, there are gaps in the coverage of the APs. Jobs may only be downloaded and uploaded under the coverage of APs.

The objective of the research in this thesis is to study the job completion time and the throughput of the VCs under different service strategies in various environments.

1.8. Contributions

The main contributions of this thesis are centered around the performance modeling of vehicular cloud systems using stochastic methods. These contributions are as follows:

1. Job completion time in a dynamic VC on a highway: In this work, we developed a stochastic model to analyze the job completion time in a VC system deployed on a highway. Our model accounts for the dynamic nature of the VC, where vehicles are constantly entering and exiting the system. The computing power and the complexity of a task are characterized by the time taken to complete the task.

   Publication:

   a. C. Tran and M. Mehmet-Ali, “Analysis of Job Completion Time in Vehicular Cloud Under Concurrent Task Execution,” International Conference on Computing,

Networking and Communications (ICNC 2023). IEEE, Feb. 22, 2023. doi: 10.1109/ICNC57223.2023.10074524

b. C. Tran and M. Mehmet-Ali, "A Performance Modeling of Dynamic Vehicular Clouds: Job Completion Time of Concurrently Executed Tasks," 6th International Workshop on Vehicular Networking and Intelligent Transportation Systems (VENITS). July 18 – *Accepted*

2. Job completion time in VC with interruption avoidance strategy: We also developed a model to analyze the job completion time in a VC system with an interruption avoidance strategy. This strategy aims to prevent disruptions to the VC by only assigning the task to a vehicle if it can complete it before it leaves the VC.

   Publication:

   a. C. Tran and M. Mehmet-Ali, "Towards Job Completion Time in Vehicular Cloud by Overcoming Resource Volatility," 2022 IEEE 47th Conference on Local Computer Networks (LCN). IEEE, Sep. 26, 2022. doi: 10.1109/lcn53696.2022.9843398.

   b. C. Tran and M. Mehmet-Ali, "Performance Analysis of Vehicular Cloud Under Interruption Avoidance Strategy," 2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). IEEE, Sep. 18, 2022. doi: 10.1109/ccece49351.2022.9918500.

3. The computing capacity of a VC with interruption avoidance strategy: In addition to analyzing the job completion time, we also evaluated the number of jobs consisting of a random number of tasks that a VC system can complete during its lifetime with an interruption avoidance strategy.

   Publication:

   a. C. Tran and M. Mehmet-Ali, "Towards Modeling Computation Capacity of a Vehicular Cloud while Overcoming Resource Volatility," 2023 Biennial Symposium on Communications (BSC 2023). IEEE, Jul. 4-7, 2023 - *Accepted*

4. The computing capacity of a robotaxi fleet: Finally, we studied the computing capacity of a robotaxi fleet, a type of VC system where self-driving vehicles provide on-demand transportation services and during their idles times perform task computations. More specifically, the average number of completed tasks that a robotaxi fleet can complete is analyzed, which can be determined numerically.

Publication:

b. C. Tran and M. Mehmet-Ali, “Robotaxis as Computing Clusters: A Stochastic Modeling Approach,” Sixteen International Workshop on Selected Topics in Mobile and Wireless Computing. IEEE, Jun. 21-23, 2023 - *Accepted*

Overall, this thesis makes measurable contributions to VC performance modeling by comprehensively analyzing the job completion time and computing capacity of VC systems under various scenarios and conditions.

1.9. Organization

The remainder of this thesis is structured as follows: Chapter 2 analyses job completion time in a dynamic VC under free-flow and congested traffic conditions with a service interruption strategy. Chapter 3 analyzes job completion time under an interruption avoidance service strategy. Chapter 4 derives the computing capacity of a VC under the interruption avoidance strategy, including the number of completed jobs for a given number of tasks. Chapter 5 derives the computing capacity of a robotaxi fleet, including the average number of tasks that the fleet can complete. Finally, Chapter 6 presents the conclusions.

Chapter 2

Job completion time in a VC with service interruption

In this chapter, we will evaluate the performance of a dynamic VC formed by moving vehicles. We will consider a VC on a highway and study its performance both under free-flow and congested traffic. In the model, the vehicles join the VC randomly, and after spending a random amount of time, they depart. Thus the population size of the VC is time-varying. We analyze the completion time of a job consisting of multiple tasks executed concurrently. We consider a task assignment scheme where a task may only be served at any time by a single vehicle, and a vehicle can only serve one task at a time. If a vehicle serving a task leaves the VC, that task must be migrated to another vehicle. The migration occurs immediately if there is an idle vehicle in the VC. Otherwise, the VC suspends the task until another vehicle becomes available. Regardless, the VC migrates the image of the task to the new vehicle to resume execution from where it was left. The execution of a job is completed when the execution of all its tasks completes.

2.1 Free-flow traffic on a highway

Traditionally, highways are considered uninterrupted flow facilities [115]. However, it is common to see congestion on highways in major cities. In this section, we will consider highways with free-flow traffic where the residency time of the vehicles on the highway is independent of each other.

First, we will analyze job completion time by assuming that migration overhead is negligible. Then we relax this assumption, introduce migration overhead to the model, and use an ad-hoc method to derive the bounds of the job completion time for this case. The main

contributions of this section are below:

- The system is modeled as a two-dimensional birth-death process, and a set of differential difference-equations describing the system's behavior is provided. After solving these equations, the job completion time's probability density function (pdf) is determined.
- Based on the first result, the probability distribution of the number of service interruptions of a task is subsequently provided.
- Finally, the upper bound for average job completion time with migration overhead is derived based on the previous results.

2.1.1. System model

Vehicle Cloud Model

We consider a highway scenario where the vehicles can wirelessly interconnect to form a VC, as shown in Fig. 2.1.

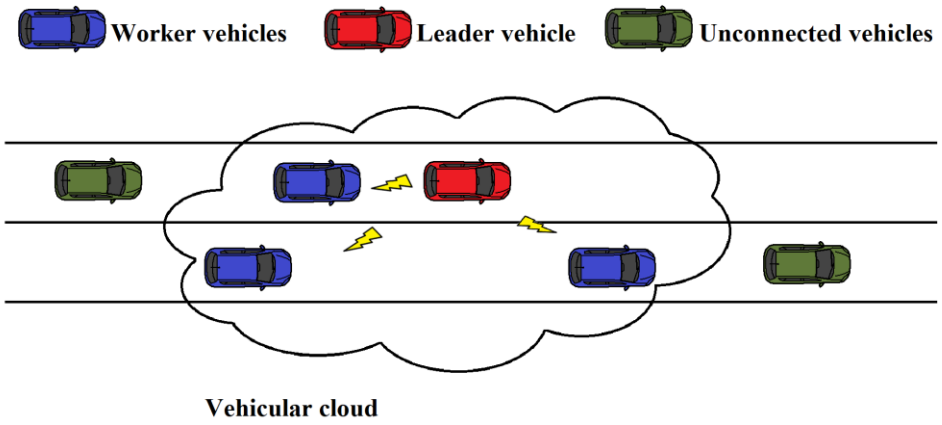


Fig. 2.1. System model

It is assumed that a vehicle may generate a new job with several tasks. This vehicle is called the owner of the job. Then, this vehicle initiates the formation of a VC and becomes the VC's leader. We further assume that all the vehicles within the transmission range of the leader become members of the VC, and these vehicles are called worker vehicles. The worker vehicles' primary function is to execute the job's tasks. A worker vehicle has two states which are referred to as active and idle. A worker vehicle is in the active state when working on a task; otherwise, it

is in the idle state. The leader only manages the assignment and migration of the tasks and does not actively process the tasks. A worker can only process a single task at any time. The size of the VC refers to the number of vehicles currently in the cloud, and it does not include the leader. The population of the VC is dynamic since the workers join to and depart from the VC as they may have different speeds than the leader. The VC terminates when processing of all the tasks of the job is completed. In this work, *workers* and *worker vehicles* will be used interchangeably. Similarly, *the leader* and *leader vehicle* are also interchangeable.

Job Model

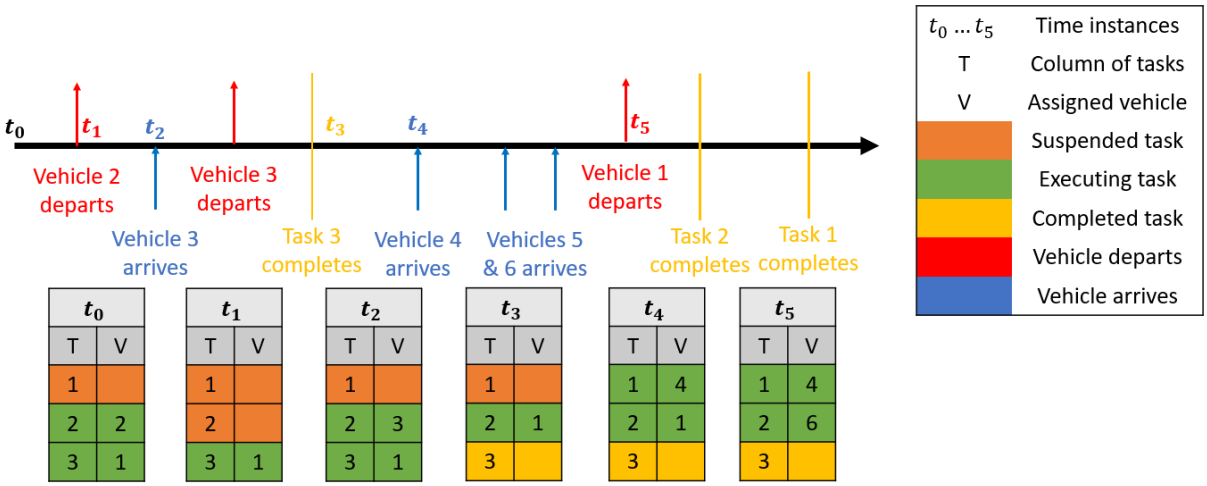
We assume that a job consists of $\mathfrak{J}$ tasks, each requiring a random amount of processing time. A worker vehicle can only execute a single task at any time and vice versa. Each task can be in three states which will be referred to as completed, active, and suspended. A task is in the completed state if its processing is complete. A task is in the active state if it is currently under execution. A task is suspended if its processing is halted because no worker can process it. In other words, the tasks in either active or suspended states are uncompleted. The processing of a job finishes when all its tasks complete their processing. During the execution of a job, job size refers to the total number of uncompleted tasks. At any time, the job size can be either bigger than, smaller than, or equal to the VC size. If the VC size is larger than the job size, it means all uncompleted tasks are in active state, and there are idle workers. On the other hand, if the VC size is smaller than the job size, the number of tasks in the active state equals to the VC size, and the remaining uncompleted tasks are in the suspended state. Finally, when the VC size equals the job size, all the uncompleted tasks are in active state, and there are no idle workers or suspended tasks.

Task Service Strategy

As explained above, each task may be served by a single vehicle, and each vehicle can only serve a single task at any time. After generating a job and forming of the VC, the leader vehicle randomly assigns the tasks to the worker vehicles initially present in the VC. If the job size is larger than the VC size, the number of idle vehicles is zero, and unassigned tasks are suspended. When a new worker joins the VC, if there are any suspended tasks, the leader randomly chooses one of the suspended tasks and assigns it to the arriving worker vehicle for execution. When a worker completes the processing of a task, the leader assigns to the vehicle a

suspended task, if any. When a worker departs from the VC, if this worker is serving a task, that task will be suspended if there are no idle workers in the VC; otherwise, the leader will randomly assign the task to one of the idle workers.

As an example in Fig. 2.2, we show an execution of a job with 3 tasks in a VC. At the time of the VC formation, t_0, there are 2 vehicles (vehicles 1 and 2) in the VC. Vehicle 1 is randomly assigned task 3, and vehicle 2 is randomly assigned task 2. As may be seen, at time t_1 vehicle 2 leaves the VC. Since there is no other vehicle that can take over the execution of task 2, the task is suspended. At t_2, vehicle 3 arrives to the VC, and the VC randomly chooses task 2 to assign to the vehicle. As vehicle 3 leaves the VC shortly after, task 2 becomes suspended again. At t_3, vehicle 1 completes task 3. As there are 2 suspended tasks, the VC randomly chooses task 2 from the suspended tasks (tasks 1 and 2) and assigns it to the vehicle 1. At t_4, vehicle 4 arrives when only task 1 is in the suspended state. Therefore, the VC picks task 1 and assigns it to this vehicle. When vehicles 5 and 6 arrive to the VC between t_4 and t_5, as there are no suspended tasks these vehicles remain idle. Finally, at t_5, vehicle 1 stops executing task 2 and departs from the VC. Since vehicles 5 and 6 are idle, VC assigns this task to randomly chosen vehicle 6. Then, the task immediately continues its execution.



t_0	
T	V
1	
2	2
3	1

t_1	
T	V
1	
2	
3	1

t_2	
T	V
1	
2	3
3	1

t_3	
T	V
1	
2	1
3	

t_4	
T	V
1	4
2	1
3	

t_5	
T	V
1	4
2	6
3	

Fig. 2.2. An example of a timeline of a VC executing a job of 3 tasks starting from the VC's creation to the completion of the job.

The notation used in this sub-section is shown in Table 2.1.

Notation	Description	Notation	Description
$\Im$	Number of tasks in a job	θ_j	Subsystem j representing the set of states that is reachable from state $\{j,k\}$ for a fixed value of j
μ	Service rate of a task	X_j	Absorbing time of subsystem θ_j to make the transition to an absorbing state
λ	Arrival rate of vehicles	$G_{X_j}(x_j)$	Cumulative distribution function of the absorbing time X_j
α	Service rate of a vehicle	$g_{X_j}(x_j)$	Probability density function of the absorbing time X_j
$\bar{r}$	Mean residency time of a vehicle in the system	$\mathcal{G}_j(s)$	Laplace transform of the probability density function of the absorbing time X_j
$\bar{s}$	Mean execution time of a task	Y_n	Completion time of the $n'th$ task
$\bar{v}$	Average number of vehicles in the system	$\mathcal{Y}_n(s)$	Laplace transform of the probability density function of completion time of the $n'th$ task
k	Number of vehicles in the system, $k \geq 0$	$\bar{Y}_n$	Average completion time of $n'th$ task
j	Number of uncompleted tasks in the system, $0 \leq j \leq \Im$	$\bar{Y}$	Average task service time
$P_j(t)$	Probability that there are j uncompleted tasks in the subsystem θ_j at time t.	γ	Probability that an execution of a task will be interrupted
$P_{j,k}(t)$	Probability that there are j uncompleted tasks and k vehicles in the subsystem θ_j at time t	ℓ	Number of interruptions that a task experiences during its service time
Q_k	Steady-state probability that there are k vehicles in the system	$\bar{\ell}$	Average number of interruptions leading to workload migration.
$\mathcal{L}_{j,k}(s)$	Laplace transform of $P_{j,k}(t)$		

Table 2.1. Main notations of section 2.1

2.1.2. An analysis of job completion time

Next, we analyze the system under the service strategy described in the previous subsection. The objective of the analysis is determining the probability density functions of task completion and job completion times. This objective requires the knowledge of the transient behavior of the system starting from the time that the job is created to the completion of the last task. Since, in this strategy, there will be service interruptions due to premature departures of workers during the task processing, we will also be interested in the probability distribution of the number of service interruptions a task undergoes during its execution.

i) Mathematical Assumptions

As stated in the previous subsection, a new job will have $\mathfrak{J}$ tasks. We assume that the execution times of the tasks are independent and identically distributed (i.i.d) according to an exponential distribution with parameter μ. The assumption of exponential execution times for tasks is often used in cloud computing [116]–[119] and, more recently, in vehicular edge computing [99]. It is shown in [120] that the distances between two adjacent vehicles on a highway are i.i.d with an exponential distribution. This result assumes a Poisson arrival of the vehicles to an arbitrary point on a highway, which was empirically confirmed in [115]. In [6], the vehicles' arrival and departure processes and the number of vehicles in a VC have been shown empirically to follow a Poisson process. As a result, we will model VC population size as a $M/M/\infty$ queueing system with a Poisson arrival process with parameter λ and exponentially distributed residency times with parameter α. We assume that at the time that a leader vehicle creates the job, the number of worker vehicles in the VC is at a steady state. Defining k as the number of worker vehicles in the VC, it is given by the steady-state probability distribution of the number of customers in a $M/M/\infty$ queueing system. From [121],

$$Q_k = Pr(k\ worker\ vehicles\ in\ the\ VC) = \frac{(\lambda/\alpha)^k}{k!} e^{-\lambda/\alpha}, \qquad k \geq 0 \tag{2.1}$$

ii) State Transition Diagram of The System

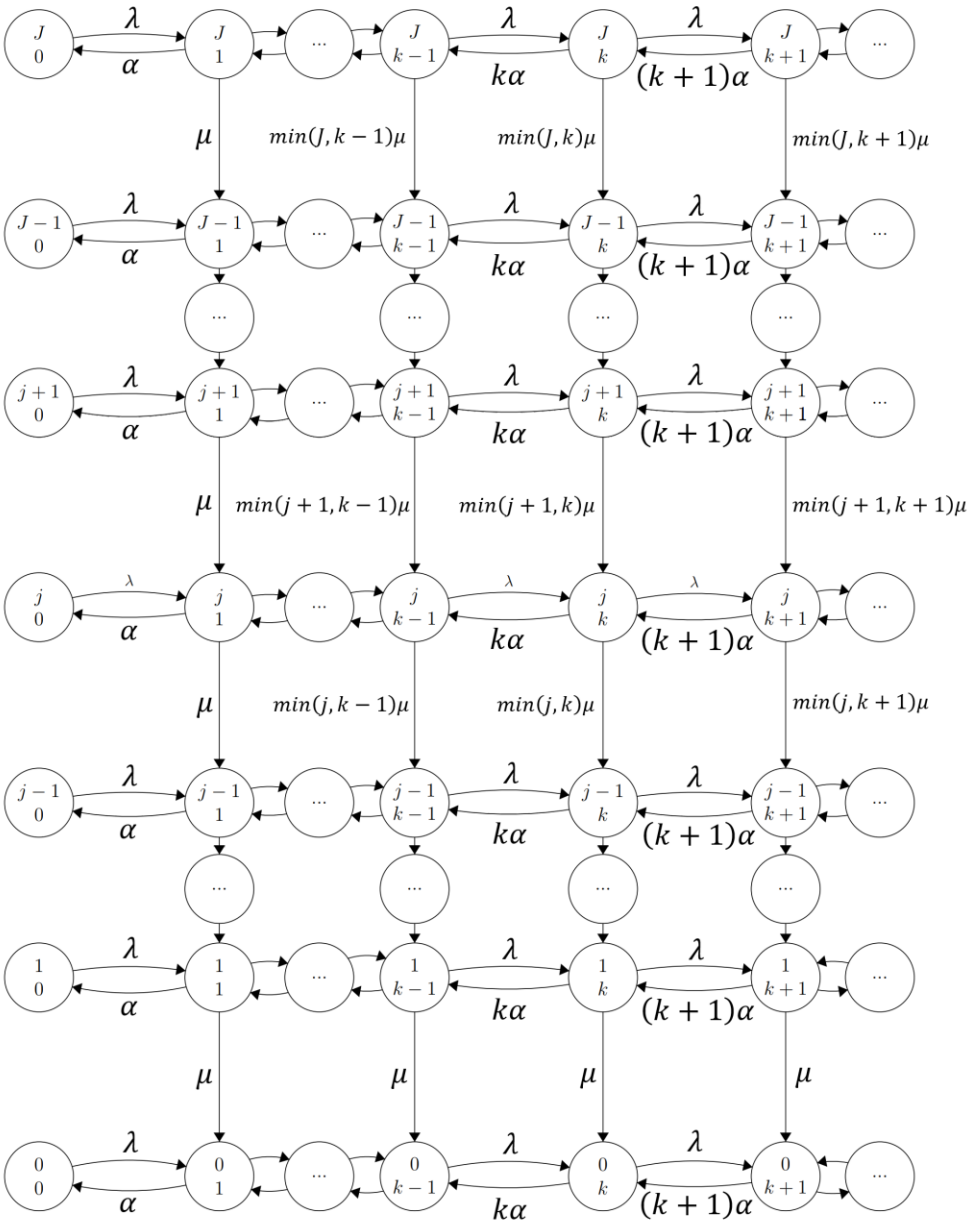


Fig. 2.3. State transition diagram for the system. In each state the number of uncompleted tasks is shown above the of number of worker vehicles in the system.

A set of two variables, $\{j, k\}$, may represent the states of the system, where j denotes the number of uncompleted tasks, and k denotes the number of worker vehicles in the system at any time. We note that when the system is in the state $\{j, k\}$, it means that the execution of $\mathfrak{J} - j$ tasks has been completed, and the number of suspended tasks is given by $\max(0, j - k)$. We can

model the system using a two-dimensional birth-death process, whose state-transition diagram is shown in Fig. 2.3. In the figure, in the states of each row the number of vehicles varies while the number of uncompleted tasks remains same. More specifically, in a row, the number of vehicles increases by rate λ as the system state moves from left to right and decreases by rate $k\alpha$, where k is the current number of vehicles in the VC, as the system state moves from right to left. In a column, the number of unfinished tasks decreases by one as the system state moves from top to bottom row by row. However, in the states of the left-most column the number of vehicles in the VC is zero, i.e., $k = 0$, as a result there are no transitions between the states in this column since there are no vehicles to execute the tasks. Additionally, in the columns that have transitions between neighboring states these transitions are uni-directional (going from top to bottom) because no new tasks are created during job execution.

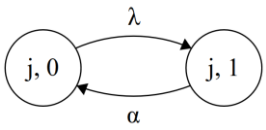
iii) Derivation of the Differential-difference equations describing the system

Next, we write the differential-difference equations describing the system in Fig. 2.3. We identify four types of states in Fig. 2.3, which are shown in Fig. 2.4.

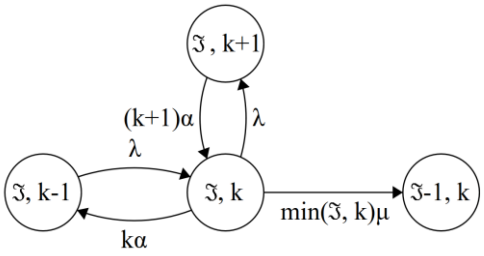
Fig. 2.4a shows the state transition diagram for the states $\{j, 0\}$ where $0 \leq j \leq \mathfrak{J}$. This subfigure represents all the states in the first column and their adjacent states in the second column in Fig. 2.3. In these states, no worker vehicles are available to process the tasks. Thus, all the unfinished tasks are in a suspended state. Therefore, the transitions between the states only involves the change in the number of vehicles in the system.

Fig. 2.4b shows the state transition diagram for the states $\{\mathfrak{J}, k\}$ where $k > 0$. This subfigure represents all the states in the first row except the left-most state and the adjacent states in the second row from the top in Fig. 2.3. In these states, none of the tasks have yet been completed, and $\min(\mathfrak{J}, k)$ of the tasks are in the active state and the remainder are in the suspended state. As previously noted, the transition between state $\{\mathfrak{J}, k\}$ and $\{\mathfrak{J} - 1, k\}$ is uni-directional since no new tasks are created during the job execution.

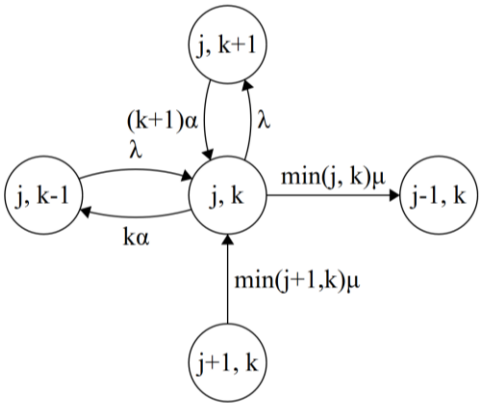
Fig. 2.4c shows the state transition diagram for the states $\{j, k\}$ where $0 < j < \mathfrak{J},\ k > 0$. The subfigure corresponds to all the states in Fig. 2.3 except the first column and the first and last rows. In these states, $\mathfrak{J} - j$ tasks are completed, $\min(j, k)$ tasks are in the active state and $j - \min(j, k)$ are in the suspended state. More specifically, we can see two uni-directional transitions from the state with an additional uncompleted task to the state with one fewer



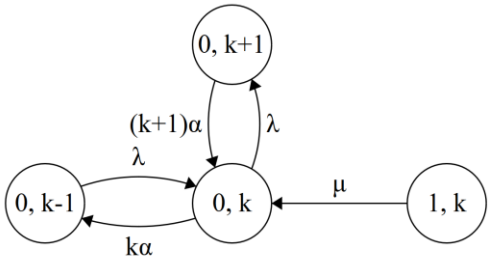
(a) State transition diagram for $k = 0, 0 \leq j \leq \Im$



(b) State transition diagram for $k > 0, j = \Im$



(c) State transition diagram for $k > 0, 0 < j < \Im$



(d) State transition diagram for $k > 0, j = 0$

Fig. 2.4. State transition diagram for the system for different values of j and k.

uncompleted tasks. Furthermore, we can see the transitions between the states on the same row as the number of vehicles in the system increases and decreases.

Fig. 2.4d shows the state transition diagram for the states $\{0, k\}$ where $k > 0$. This subfigure corresponds to all the states on the bottom row and their adjacent states above in Fig. 2.3. As the last row represents states with no uncompleted tasks remaining in the system (or all tasks have been completed), there is only a single uni-directional transition into states $\{0, k\}$.

Next, let $\mathfrak{P}_{j,k}(t)$ denote the probability that there will be j uncompleted tasks and k worker vehicles in the system at time t. Then, from the state transition diagrams in Fig. 2.4, we can write the following differential-difference equations describing the system,

$$\frac{d\mathfrak{P}_{j,0}(t)}{dt} = \alpha\mathfrak{P}_{j,1}(t) - \lambda\mathfrak{P}_{j,0}(t), \qquad \Im \geq j \geq 0, k = 0 \tag{2.2}$$

$$\frac{d\mathfrak{P}_{\mathfrak{J},k}(t)}{dt} = \lambda\mathfrak{P}_{\mathfrak{J},k-1}(t) + (k+1)\alpha\mathfrak{P}_{\mathfrak{J},k+1}(t) - [\lambda + k\alpha + min(\mathfrak{J},k)\mu]\mathfrak{P}_{\mathfrak{J},k}(t), \qquad j = \mathfrak{J}, k > 0 \tag{2.3}$$

$$\frac{d\mathfrak{P}_{j,k}(t)}{dt} = \lambda\mathfrak{P}_{j,k-1}(t) + (k+1)\alpha\mathfrak{P}_{j,k+1}(t) - [\lambda + k\alpha + min(j,k)\mu]\mathfrak{P}_{j,k}(t) + min(j+1,k)\mu\mathfrak{P}_{j+1,k}(t), \qquad 0 < j < \mathfrak{J}, \qquad k > 0 \tag{2.4}$$

$$\frac{d\mathfrak{P}_{0,k}(t)}{dt} = \lambda\mathfrak{P}_{0,k-1}(t) + (k+1)\alpha\mathfrak{P}_{0,k+1}(t) - [\lambda + k\alpha]\mathfrak{P}_{0,k}(t) + \mu\mathfrak{P}_{1,k}(t), \qquad j = 0, \qquad k > 0 \tag{2.5}$$

The above set of differential-difference equations determines the behavior of the system, and the solution of the set of equations must satisfy the normalization condition,

$$\sum_{j=0}^{\mathfrak{J}}\sum_{k=0}^{\infty}\mathfrak{P}_{j,k}(t) = 1 \tag{2.6}$$

Next, we let $\mathfrak{P}_j(t)$, $\mathfrak{P}_k(t)$ denote the marginal probability distributions of the number of uncompleted tasks and the number of worker vehicles in the system at time t, respectively. Then,

$$\mathfrak{P}_j(t) = \sum_{k=0}^{\infty}\mathfrak{P}_{j,k}(t) \tag{2.7}$$

$$\mathfrak{P}_k(t) = \sum_{j=0}^{\mathfrak{J}}\mathfrak{P}_{j,k}(t) \tag{2.8}$$

$$\mathfrak{P}_j(0) = \begin{cases} 1 & for\ j = \mathfrak{J} \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

$$\mathfrak{P}_k(t) = Q_k, \qquad k \geq 0 \tag{2.10}$$

where in (2.10), Q_k is given by (2.1). We note that if the initial size of VC is larger than the number of tasks in the job, all the tasks will be initially under execution by the worker vehicles.

In Appendix A, we have attempted to solve the above set of differential-difference equations through transform methods. However, this approach was not successful because we could not determine all the unknowns. As a result, we tried the decomposition approach to below.

iv) Derivation of the joint probability distribution of the number of uncompleted tasks and vehicles through decomposition

In the state transition diagram of Fig. 2.3, the states in row j correspond to the system having j uncompleted tasks. When the system is in row j, it moves to the row $j-1$ immediately when one of the tasks completes its execution. The execution of the job begins in one of the states at the top row $\Im$, and it's completed when the system enters one of the states at the bottom row. Therefore, the job completion time is the sum of the periods the system spends in the states in each row. Following this observation, we will derive the amount of time the system spends in the states of each row. As a result, we will divide the system into several subsystems, where we can analyze each subsystem independently of the other subsystems. Let θ_j denote the subsystem j where $0 < j \leq \Im$. We define the subsystem θ_j as the set of states of rows j and $j-1$ in Fig. 2.3 with the state-transition diagram as shown in Fig. 2.5.

$$\theta_j = \{(j,k)\ \forall\ k \geq 0\ \cup (j-1,k)\ \forall\ k > 0\}, \qquad 0 < j \leq \Im \tag{2.11}$$

The states of row $j-1$ for $k > 0$ will be absorbing states [121] for this subsystem. When the subsystem enters one of the absorbing states, then the system immediately exits the subsystem θ_j and enters the subsystem θ_{j-1}. This transition happens when service of one of the tasks is completed. We will refer to the amount of time that the system spends in the subsystem θ_j as absorbing time and will denote it by X_j.

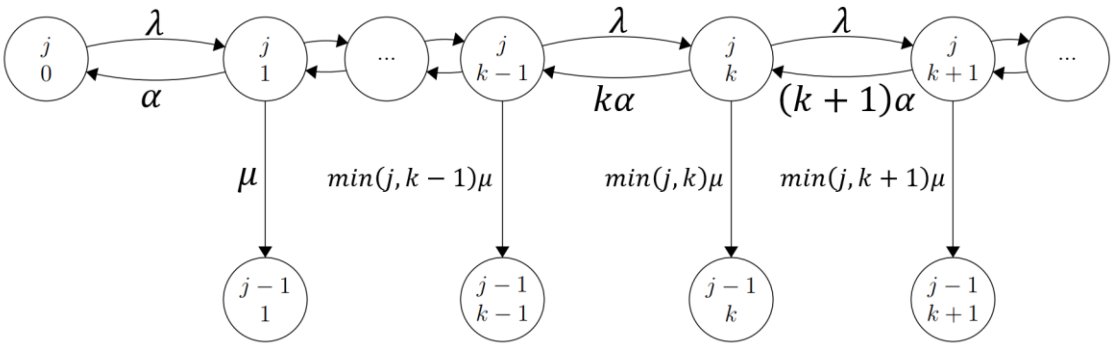


Fig. 2.5. State transition diagram for subsystem θ_j. States $\{j-1, k\}$ are absorbing states.

Next, we will derive the joint probability distribution of the number of uncompleted tasks and the number of vehicles in a subsystem as a function of time. Since the subsystems are analyzed independently, we set the initial time for each subsystem to zero. Let $P_{j,k}(t)$ denote the probability that there will be j uncompleted tasks and k worker vehicles in the subsystem θ_j at time t. From Fig. 2.5, the behavior of the subsystem θ_j may be described by the following set of differential-difference equations,

$$\frac{dP_{j,k}(t)}{dt} = \lambda P_{j,k-1}(t) + (k+1)\alpha P_{j,k+1}(t) - [\lambda + k\alpha + min(j,k)\mu]P_{j,k}(t), \qquad k > 0 \tag{2.12}$$

$$\frac{dP_{j-1,k}(t)}{dt} = min(j,k)\mu P_{j,k}(t), \qquad k > 0 \tag{2.13}$$

$$\frac{dP_{j,0}(t)}{dt} = \alpha P_{j,1}(t) - \lambda P_{j,0}(t), \qquad k = 0 \tag{2.14}$$

Since the number of worker vehicles is at the steady state, the initial distribution of the number of workers in the subsystem $\theta_{\mathfrak{J}}$ is given by (2.1). In the subsystem θ_j, $0 < j < \mathfrak{J}$, the initial number of vehicles in the subsystem will be given by the number of vehicles in the subsystem θ_{j+1} when that subsystem enters an absorbing state. As a result, we have the following initial distributions,

$$P_{j,k}(0) = \begin{cases} Q_k, & for\ k \geq 0\ and\ \theta_j, j = \Im \\ P_{j+1,k}(\infty), & for\ \ \theta_j, j < \Im \end{cases} \tag{2.15}$$

Moreover, we also note that at time $t = 0$, the VC cannot have any completed task in the subsystem θ_j, thus, we can write,

$$P_{j-1,k}(0) = 0, \qquad 0 < j \leq \Im, \qquad k > 0 \tag{2.16}$$

Next, let us define the following Laplace transform,

$$\mathcal{L}_{j,k}(s) = \mathbb{L}\{P_{j,k}(t)\} = \int_{t=0}^{\infty} P_{j,k}(t)\ e^{-st} dt \tag{2.17}$$

To solve the set of equations (2.12) to (2.14), we will take their Laplace transforms,

$$s\mathcal{L}_{j,k}(s) - P_{j,k}(0) = \lambda\mathcal{L}_{j,k-1}(s) + (k+1)\alpha\mathcal{L}_{j,k+1}(s) - [\lambda + k\alpha + min(j,k)\mu]\mathcal{L}_{j,k}(s)$$

$$k > 0 \tag{2.18}$$

$$s\mathcal{L}_{j-1,k}(s) - P_{j-1,k}(0) = min(j,k)\mu\mathcal{L}_{j,k}(s), \qquad k > 0 \tag{2.19}$$

$$s\mathcal{L}_{j,0}(s) - P_{j,0}(0) = \alpha\mathcal{L}_{j,1}(s) - \lambda\mathcal{L}_{j,0}(s), \qquad k = 0 \tag{2.20}$$

Next, we express $\mathcal{L}_{j,k}(s)$ in terms of $\mathcal{L}_{j,0}(s)$. From (2.20), we can write $\mathcal{L}_{j,1}(s)$ in terms of $\mathcal{L}_{j,0}(s)$,

$$\mathcal{L}_{j,1}(s) = \frac{1}{\alpha}[(s+\lambda)\mathcal{L}_{j,0}(s) - P_{j,0}(0)] \tag{2.21}$$

Substituting $k = 1$ in (2.18) while noting $j \geq 1$ gives,

$$s\mathcal{L}_{j,1}(s) - P_{j,1}(0) = \lambda\mathcal{L}_{j,0}(s) + 2\alpha\mathcal{L}_{j,2}(s) - [\lambda + \alpha + \mu]\mathcal{L}_{j,1}(s)$$

$$\Leftrightarrow \mathcal{L}_{j,2}(s) = \frac{[s + \lambda + \alpha + \mu]\mathcal{L}_{j,1}(s) - \lambda\mathcal{L}_{j,0}(s) - P_{j,1}(0)}{2\alpha} \tag{2.22}$$

Substituting $\mathcal{L}_{j,1}(s)$ in (2.21) to (2.22), we can also express $L_{j,2}(s)$ in terms of $L_{j,0}(s)$ as follows:

$$\mathcal{L}_{j,2}(s) = \frac{[s+\lambda+\alpha+\mu]\frac{1}{\alpha}[(s+\lambda)\mathcal{L}_{j,0}(s) - P_{j,0}(0)] - \lambda\mathcal{L}_{j,0}(s) - P_{j,1}(0)}{2\alpha}$$

$$= \frac{\left[\frac{1}{\alpha}(s+\lambda+\alpha+\mu)(s+\lambda) - \lambda\right]\mathcal{L}_{j,0}(s) - [s+\lambda+\alpha+\mu]\frac{1}{\alpha}P_{j,0}(0) - P_{j,1}(0)}{2\alpha} \quad (2.23)$$

Next, we solve for $L_{j,k+1}(s)$ in (2.18),

$$\mathcal{L}_{j,k+1}(s) = \frac{[s+\lambda+k\alpha+min(j,k)\mu]}{(k+1)\alpha}\mathcal{L}_{j,k}(s) - \frac{\lambda}{(k+1)\alpha}\mathcal{L}_{j,k-1}(s) - \frac{1}{(k+1)\alpha}P_{j,k}(0),$$

$$for\ k > 0 \quad (2.24)$$

In (2.24), we see that $\mathcal{L}_{j,k+1}(s)$ depends on $\mathcal{L}_{j,k}(s)$, and $\mathcal{L}_{j,k-1}(s)$. As a result, we can express $\mathcal{L}_{j,k+1}(s)$ recursively in terms of $\mathcal{L}_{j,0}(s)$ using (2.21) and (2.22). This implies $\mathcal{L}_{j,k}(s)$ can be written as a function of $\mathcal{L}_{j,0}(s)$ for $k > 0$. Next, we show how to determine $\mathcal{L}_{j,0}(s)$ from the normalization condition,

$$\sum_{k=0}^{\infty} P_{j,k}(t) + \sum_{k=1}^{\infty} P_{j-1,k}(t) = 1 \quad (2.25)$$

Taking Laplace transform of the conservation relation in the above, we have,

$$\sum_{k=0}^{\infty} \mathcal{L}_{j,k}(s) + \sum_{k=1}^{\infty} \mathcal{L}_{j-1,k}(s) = \frac{1}{s} \quad (2.26)$$

After substituting (2.16) in (2.19), we solve for $\mathcal{L}_{j-1,k}(s)$ as followed:

$$\mathcal{L}_{j-1,k}(s) = \frac{min(j,k)\mu}{s}\mathcal{L}_{j,k}(s), \qquad k > 0 \quad (2.27)$$

Then, results in (2.21) and (2.27) are substituted on the left-hand side of (2.26), which gives,

$$\sum_{k=0}^{\infty} \mathcal{L}_{j,k}(s) + \sum_{k=1}^{\infty} \frac{min(j,k)\mu}{s} \mathcal{L}_{j,k}(s)$$

$$= s\mathcal{L}_{j,0}(s) + \sum_{k=1}^{\infty} s\mathcal{L}_{j,k}(s) + \sum_{k=1}^{\infty} min(j,k)\mu \mathcal{L}_{j,k}(s)$$

$$= s\mathcal{L}_{j,0}(s) + \sum_{k=1}^{\infty} [s + min(j,k)\mu]\mathcal{L}_{j,k}(s)$$

$$= s\mathcal{L}_{j,0}(s) + (s+\mu)\mathcal{L}_{j,1}(s) + \sum_{k=2}^{\infty} [s + min(j,k)\mu]\mathcal{L}_{j,k}(s)$$

$$= s\mathcal{L}_{j,0}(s) + (s+\mu)\frac{1}{\alpha}[(s+\lambda)\mathcal{L}_{j,0}(s) - P_{j,0}(0)]$$

$$+ \sum_{k=2}^{\infty} [s + min(j,k)\mu]\mathcal{L}_{j,k}(s) \tag{2.28}$$

Next, using the change of variables, we can rewrite equation (2.24) as below:

$$\mathcal{L}_{j,k}(s) = \frac{[s + \lambda + (k-1)\alpha + min(j,k-1)\mu]}{k\alpha} \mathcal{L}_{j,k-1}(s) - \frac{\lambda}{k\alpha} \mathcal{L}_{j,k-2}(s)$$

$$- \frac{1}{k\alpha} P_{j,k-1}(0),$$

$$for\ k \geq 2 \tag{2.29}$$

Since $P_{j,k}(0)\ \forall k \geq 1$ can be found from (2.15), as previously noted, $\mathcal{L}_{j,k}(s)$ can be expressed as a function of only $\mathcal{L}_{j,0}(s)$ from the recursion in (2.29). As a result, the sum $\sum_{k=2}^{\infty}[s + min(j,k)\mu]\mathcal{L}_{j,k}(s)$ in (2.28) can be expressed as a function of $\mathcal{L}_{j,0}(s)$. Therefore, we can solve for the unknown function $\mathcal{L}_{j,0}(s)$ using (2.26).

After the determination of $\mathcal{L}_{j,0}(s)$, this means that we have obtained all $\mathcal{L}_{j,k}(s)$ functions using (2.21), (2.28) and (2.26). Then, by taking the inverse Laplace transforms of $\mathcal{L}_{j,k}(s)$, $P_{j,k}(t)$

can be finally derived. In determining $\mathcal{L}_{j,0}(s)$ we need to truncate the infinite summation in (2.28), the accuracy of this truncation is tested through simulation.

We note that from the final value theorem property of the Laplace transforms,

$$P_{j+1,k}(\infty) = \lim_{s\to 0} s\mathcal{L}_{j+1,k}(s) \tag{2.30}$$

which gives the initial distribution of the number of vehicles in the subsystem θ_j, $0 < j < \Im$, in (2.15). This initial distribution of the number of vehicles in a subsystem has also been confirmed by simulation.

v) Derivation of the probability density function of the job completion time

We define the service time of a task as the time interval between the job generation and the task completion time. Let τ_i, s_i denote the execution and service time of the task i, respectively. Also, let sets T, S denote the execution and service times of all the tasks in the job, respectively,

$$T = \{\tau_1, \tau_2, \dots, \tau_i, \dots., \tau_\Im\}$$

$$S = \{s_1, s_2, \dots., s_i, \dots., s_\Im\}$$

We note that the order of the service times of the tasks is not necessarily the same as their execution time durations since the random assignment of the tasks to the vehicles and suspension of the tasks may change the order of the service time completions. Thus, it is possible to have $s_i < s_j$ despite of $\tau_i > \tau_j$, $1 \le i, j \le \Im$. Let us define Y as the ordered set of task service times,

$$Y = \{Y_1, Y_2, \dots, Y_n, \dots, Y_\Im\}$$

where Y_1 denote the smallest and $Y_\Im$ the largest of the task service times. Thus, the elements of the ordered set of task service times satisfy the following inequalities,

$$Y_1 < Y_2 < \cdots Y_n < \cdots < Y_\Im \tag{2.31}$$

As we noted before, when the subsystem θ_j reaches an absorbing state, the service time of a task completes, and that subsystem makes a transition to the subsystem θ_{j-1}. The ordered

service times of the tasks and absorbing time of the subsystems are related by the following equations,

$$Y_1 = X_{\Im} \tag{2.32}$$

$$Y_n = Y_{n-1} + X_{(\Im - n+1)} \tag{2.33}$$

$$Y_{\Im} = Y_{\Im-1} + X_1 \tag{2.34}$$

We note that the service time of the last task to complete, $Y_{\Im}$, also corresponds to the completion time of the job. Fig. 2.6 shows an example of the relationship between X_j, τ_i, s_i and Y_n for $\Im = 5$.

First, we will determine the pdf of the absorbing time of each subsystem. Let us define the cumulative distribution function (CDF) of the absorbing time of the subsystem θ_j as,

$$\begin{aligned} G_{X_j}(x_j) &= Prob(X_j \leq x_j) \\ &= 1 - Prob(X_j > x_j) \end{aligned} \tag{2.35}$$

Let $P_j(t)$ denote the marginal probability distribution of the number of uncompleted tasks in the subsystem θ_j at time t, then this distribution is given by,

$$P_j(t) = \sum_{k=0}^{\infty} P_{j,k}(t) \tag{2.36}$$

We note that if X_j is the absorbing time of the subsystem θ_j, at moment X_j, the system will make the transition from the subsystem θ_j to subsystem θ_{j-1}. Thus, the event $X_j > x_j$ is equivalent to the event that there are j uncompleted tasks in the subsystem θ_j at time t. Then, the complementary probability $Prob(X_j > x_j)$ maybe determined from the marginal distribution of the number of tasks in the subsystem θ_j as follow:

$$Prob(X_j > x_j) = P_j(x_j) \tag{2.37}$$

Substituting the above in (2.35), we have,

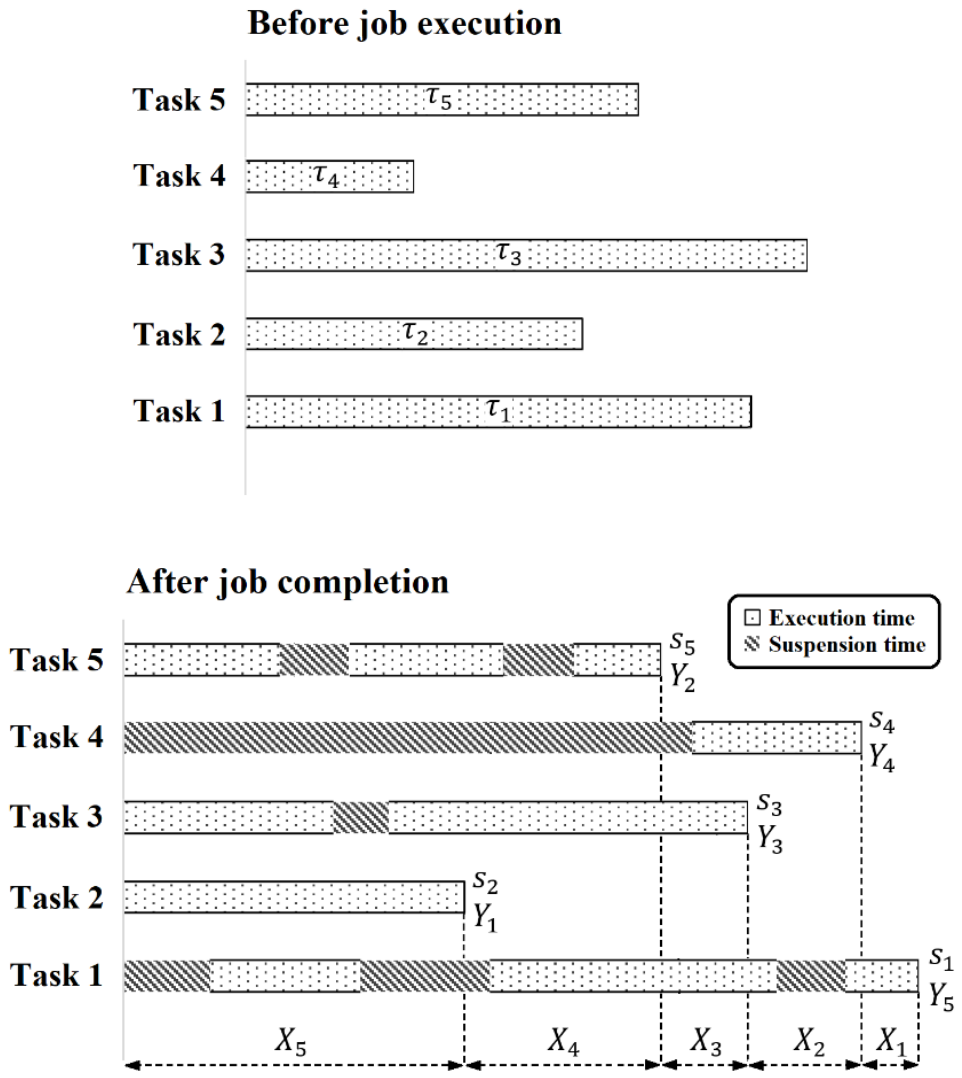


Fig. 2.6. An example showing task execution and service times, subsystem absorption times for a job with $\Im = 5$ tasks. τ_i and s_i are execution and service times of task *i*, Y_i is the service time of the task that is *i*'th to complete.

$$G_{X_j}(x_j) = 1 - P_j(x_j) \tag{2.38}$$

Then, the pdf of the absorbing time of the subsystem θ_j is given by

$$g_{X_j}(x_j) = \frac{dG_{X_j}(x_j)}{dx_j} = -\frac{dP_j(x_j)}{dx_j} \tag{2.39}$$

Next, we determine the pdfs of the completion time of the tasks. Let us define Laplace

transforms of the pdf of absorbing time of subsystem θ_j and service time of the task completed in this subsystem as,

$$\mathcal{G}_j(s) = E[e^{-sX_j}] \tag{2.40}$$

$$\mathcal{Y}_n(s) = E[e^{-sY_n}] \tag{2.41}$$

Since the absorbing times of the subsystems are independent of each other, we can first rewrite equations (2.32) to (2.34) as follow,

$$\mathcal{Y}_1(s) = \mathcal{G}_{\Im}(s) \tag{2.42}$$

$$\mathcal{Y}_n(s) = \mathcal{Y}_{n-1}(s)\mathcal{G}_{(\Im-n+1)}(s) \tag{2.43}$$

$$\mathcal{Y}_{\Im}(s) = \mathcal{Y}_{\Im-1}(s)\mathcal{G}_1(s) \tag{2.44}$$

From the above recursion, we can determine $\mathcal{Y}_n(s)$ and Laplace transform of the pdf of the job completion time, $\mathcal{Y}_{\Im}(s)$, respectively, as

$$\mathcal{Y}_n(s) = \prod_{i=\Im-n+1}^{\Im} \mathcal{G}_i(s), \quad 1 \le n \le \Im \tag{2.45}$$

$$\mathcal{Y}_{\Im}(s) = \prod_{i=1}^{\Im} \mathcal{G}_i(s) \tag{2.46}$$

Let $f_{Y_n}(t)$ denote the pdf of the random variable Y_n. Then, $f_{Y_n}(t)$ may be obtained by the inversion of $\mathcal{Y}_n(s)$. Similarly, the inversion of $\mathcal{Y}_{\Im}(s)$ gives the pdf of the job completion time $f_{Y_{\Im}}(t)$. From the above, we can also obtain, the m^{th} moment of the task service time using the following relation,

$$\bar{Y}_n^m = (-1)^m \frac{d^m \mathcal{Y}_n(s)}{ds^m}\bigg|_{s=0} \tag{2.47}$$

From the above, the first moment of service time of the $n'th$ task to be completed is given by,

$$\bar{Y}_n = -\left.\frac{d\mathcal{Y}_n(s)}{ds}\right|_{s=0} \tag{2.48}$$

The expected value of job completion time is given by $\bar{Y}_{\Im}$ using $\mathcal{Y}_{\Im}(s)$, from (2.46)

$$\bar{Y}_{\Im} = -\left.\frac{d\mathcal{Y}_{\Im}(s)}{ds}\right|_{s=0} \tag{2.49}$$

Let $\bar{Y}$ denote the average of the service times of all the tasks. Then, it is given by,

$$\bar{Y} = \frac{1}{\Im}\sum_{n=1}^{\Im} \bar{Y}_n \tag{2.50}$$

2.1.3. Derivation of the probability distribution of the number of service interruptions

In this subsection, we determine the probability distribution of the number of service interruptions of a task. Service of a task will be interrupted when a worker vehicle serving that task departs prematurely from the system. Let τ and r denote the execution time of a task and the residency time of a vehicle in the VC, respectively. Let γ denote the probability that the execution of a task will be interrupted, then,

$$\gamma = Pr(\tau > r) \tag{2.51}$$

Since the task execution times and the vehicle residency times are exponentially distributed with parameters μ and α respectively, then,

$$\begin{aligned}\gamma &= \int_0^{\infty}\int_0^{\tau} \mu e^{-\mu\tau}\alpha e^{-\alpha r} dr d\tau = \int_0^{\infty} \mu e^{-\mu\tau}\left[\int_0^{\tau} \alpha e^{-\alpha r} dr\right] dt \\ &= \int_0^{\infty} \mu e^{-\mu\tau}(1 - e^{-\alpha\tau}) dt = \int_0^{\infty} \mu e^{-\mu\tau} dt - \int_0^{\infty} \mu e^{-(\mu+\alpha)\tau} dt \\ &= \frac{\alpha}{\alpha + \mu}\end{aligned} \tag{2.52}$$

Let ℓ denote the number of interruptions a task experiences during its service time. Since

the service interruptions are independent of each other, then the probability distribution of the number of interruptions during the execution of a task is given by the geometric distribution,

$$Prob(\ell = k) = (1-\gamma)\gamma^k, \qquad k = 0, 1, 2, \ldots \tag{2.53}$$

The average number of interruptions leading to a workload migration is given by

$$\bar{\ell} = \frac{\gamma}{1-\gamma} \tag{2.54}$$

It is noted that not every service interruption results in the suspension of a task. If an idle worker vehicle is available when a task is interrupted, its service will immediately resume. However, every service interruption results in migration overhead.

2.1.4. Numerical and Simulation Results

In this subsection, we present the numerical results about the analysis and simulation results to verify the analysis. The numerical results are obtained by solving for $\mathcal{L}_{0,k}(s)$ in (2.28). This derivation requires truncation of the infinite summation. We set the upper limit of the sum to be at least twice the mean of the average number of vehicles in the VC. As will be seen, the simulation results validate this choice of the truncation limit. We used Monte Carlo simulation to validate the numerical results, test the accuracy of the truncation of the infinite summation in (2.28) and confirm that the analysis is error-free. The simulation program was written in Matlab and is described in Appendix B.

We have chosen five cases to illustrate the results based on the average number of vehicles in the VC, $\bar{v}$, the average task execution times, $\bar{\tau}$, the mean vehicle residency time, $\bar{r}$, and the number of tasks in a job, $\mathfrak{J}$. Table 2.2 presents the parameter values for each case in a column. For cases 4 and 5, the residency time of the vehicles and the average number of vehicles in the VC is higher. As a result, processing of jobs with a higher number of tasks with longer task execution times will be possible.

Case	*1*	*2*	*3*	*4*	*5*
$\bar{r}$ **(min)**	6	6	6	20	20
$\bar{v}$	2	4	8	9	9
$\bar{\tau}$**(min)**	6	6	9	20	25
$\mathfrak{J}$	2	4	4	6	8

Table 2.2. Values of system parameters

Table 2.3 presents the numerical and simulation results of the ordered average task service times for cases 2 and 3 in Table 2.2. Also shown in the last column of Table 2.3. is the overall average task service time ($\bar{Y}$). It may be seen that the overall average service time is larger than the average task execution time, $\bar{Y} > \bar{\tau}$. This difference is due to task suspension times, which results from not having an idle vehicle to continue with the execution of a task. The difference between the average task service time and the average task execution time gives the average task suspension time, $\bar{Y} - \bar{\tau}$. In Fig. 2.7, we plot the average task suspension time as a function of the average number of vehicles in the VC, $\bar{v}$, for Case 2. It may be seen that as $\bar{v}$ increases, the average task suspension time decreases and eventually drops to zero. From this point on, the job completion time is no longer dependent on the average number of vehicles in the VC. We note that the numerical and simulation results are very close in both the table and the figure.

Case		$\bar{Y}_1$	$\bar{Y}_2$	$\bar{Y}_3$	$\bar{Y}_4$	$\bar{Y}$
2	**Num**	2.101	4.409	7.569	13.67	6.937
	Sim	2.103	4.412	7.574	13.67	6.94
3	**Num**	2.293	5.311	9.817	18.82	9.06
	Sim	2.293	5.308	9.815	18.82	9.059

Table 2.3. Numerical and simulation results for average task service time in minutes for Cases 2 and 3

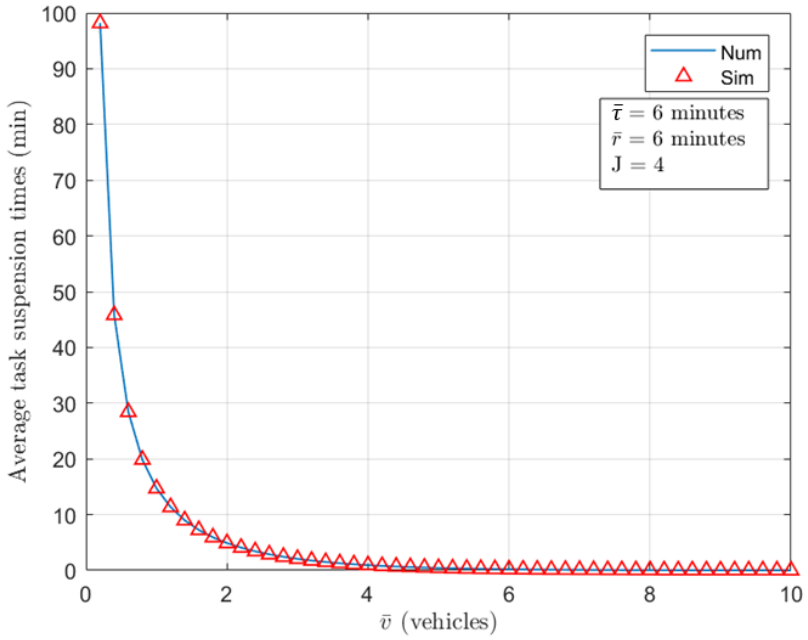


Fig. 2.7. Numerical and simulation results for average task suspension times as a function of the average number of vehicles in the VC for Case 2.

Table 2.4 presents the average job completion time for all cases shown in Table 2.2. It may be seen that the average job completion times in cases 4 and 5 are significantly higher than the other cases because their average task execution times are longer, and the number of tasks in the job is also higher. Fig. 2.8 to Fig. 2.10 show the average job completion time as a function of the average number of vehicles in the VC, $\bar{v}$, for cases 1, 3, 5 in Table 2.2. As may be seen, the average job completion times continuously decrease with increasing $\bar{v}$, until it reaches to a plateau, at which, the job completion times are seemingly no longer dependent on $\bar{v}$. As shown in Fig. 2.7, this behavior is due to the elimination of the task suspension times with increasing $\bar{v}$.

Fig. 2.11 to Fig. 2.13 show both the numerical and simulation results for the probability density function of the job completion times for cases 1, 3, and 5 in Table 2.2. From the graphs, we can see that the numerical results match the simulation results.

Case No	1	2	3	4	5
$\bar{Y}_{\mathfrak{J}}$ (num)	11.861	13.666	18.82	49.296	68.954
$\bar{Y}_{\mathfrak{J}}$ (sim)	11.869	13.669	18.818	49.282	68.956

Table 2.4. Numerical and simulation results for average job completion time in minutes for cases in Table 2.2

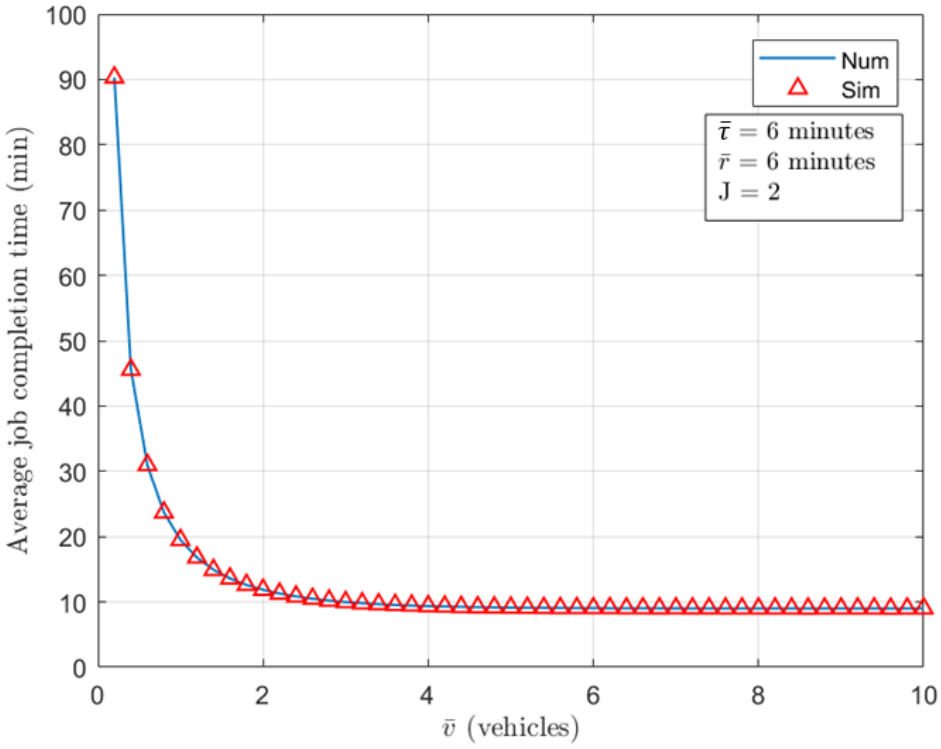


Fig. 2.8. Average job completion time as a function of the average number of vehicles in the VC for case 1in Table 2.2.

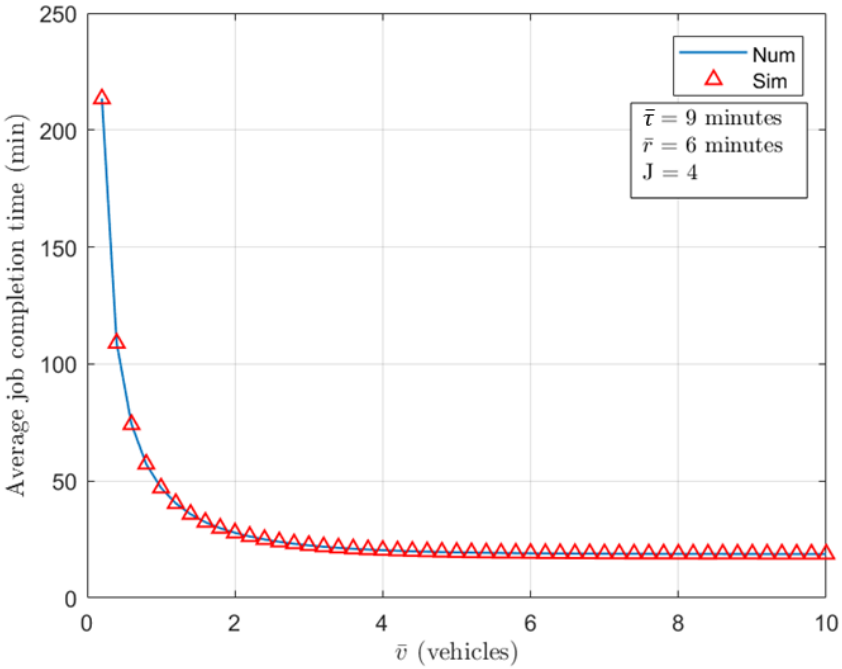


Fig. 2.9. Average job completion time as a function of the average number of vehicles in the VC for case 3 in Table 2.2.

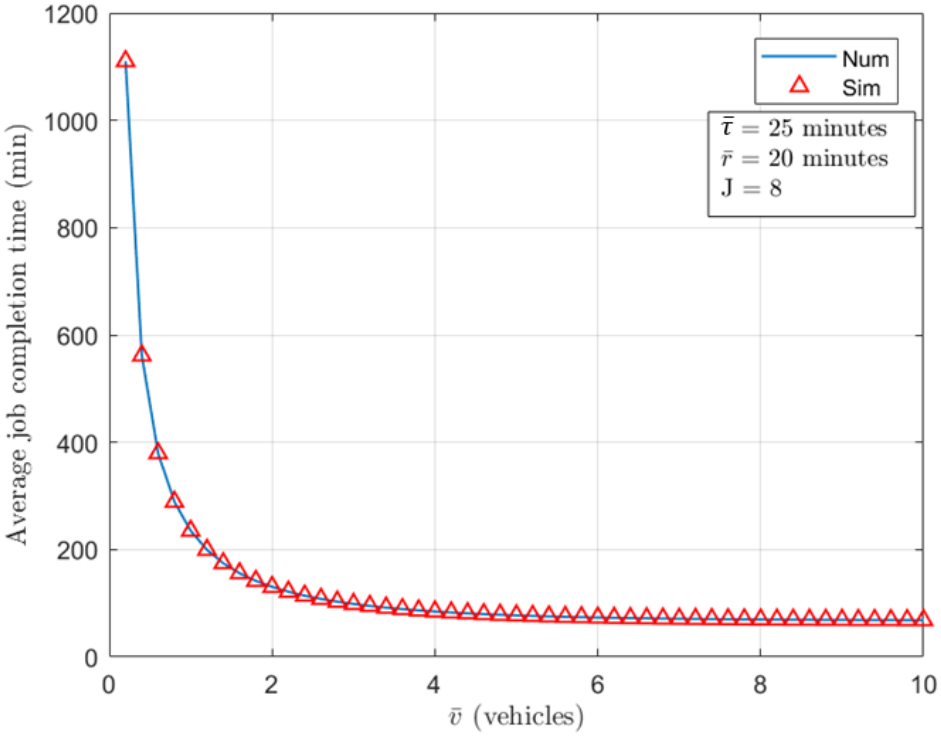


Fig. 2.10. Average job completion time as a function of the average number of vehicles in the VC for case 5 in Table 2.2.

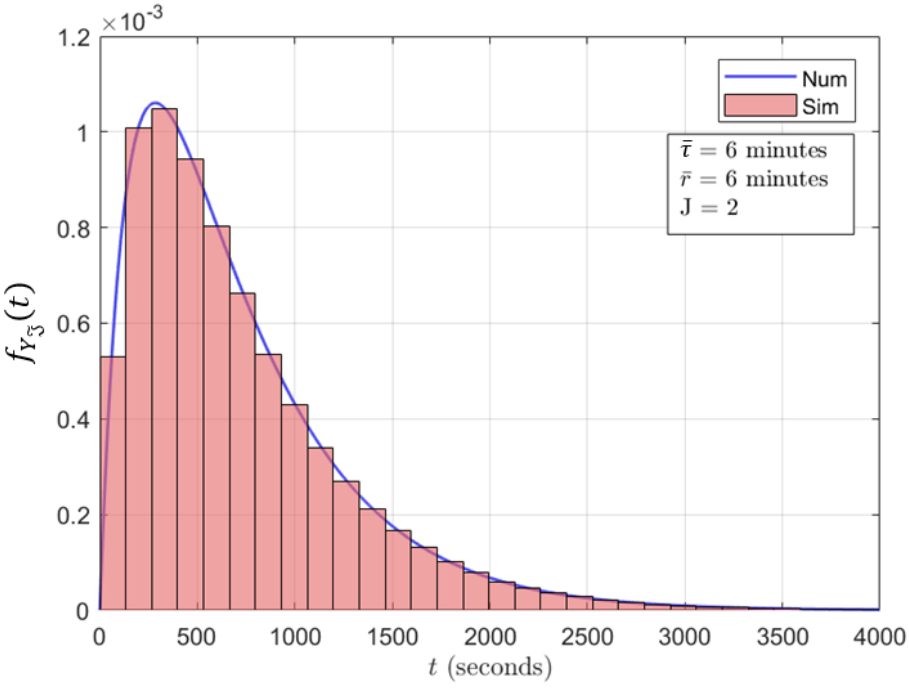


Fig. 2.11. Probability density function of the job completion time for the case 1 in Table 2.2

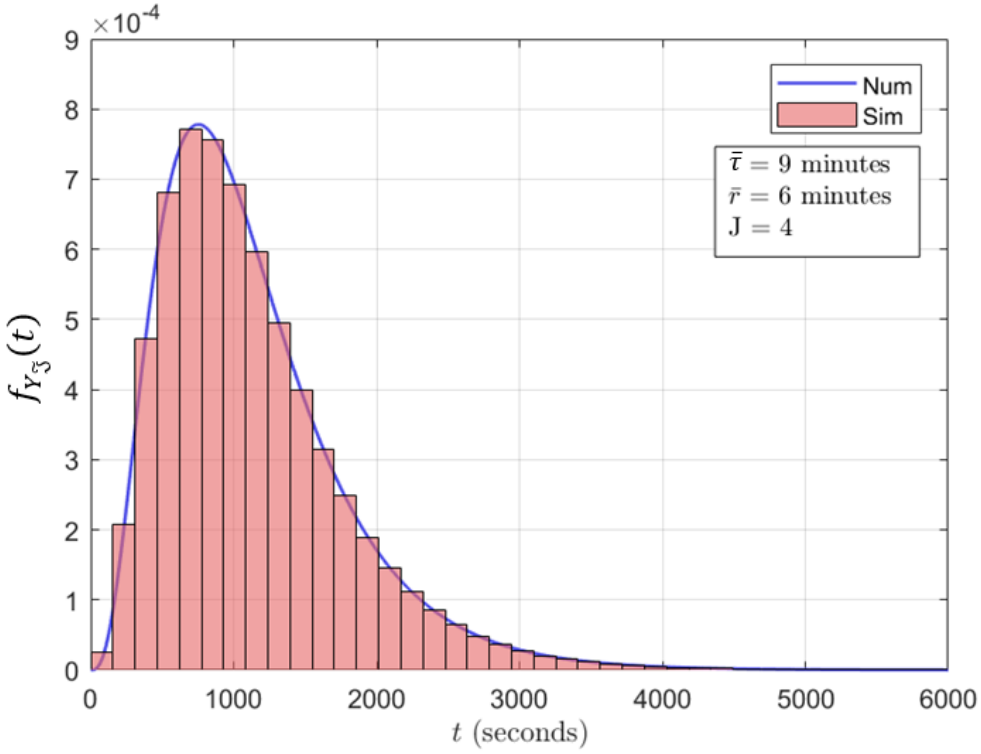


Fig. 2.12. Probability density function of the job completion time for the case 3 in Table 2.2

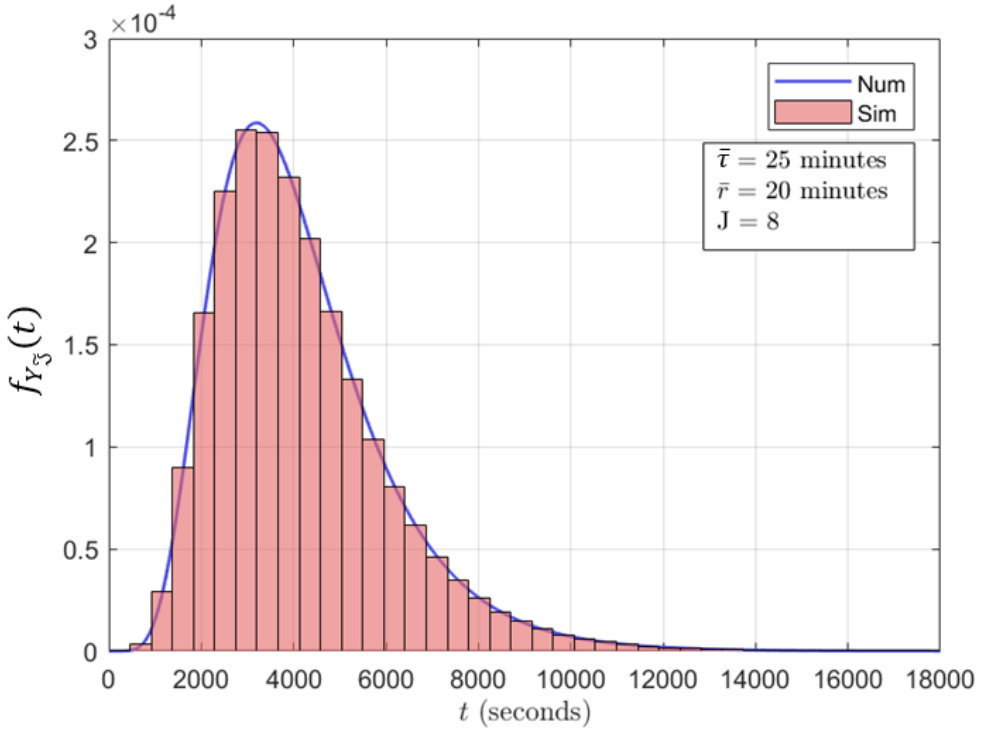


Fig. 2.13. Probability density function of the job completion time for the case 5 in Table 2.2

Fig. 2.14 shows the average number of service interruptions of a task as a function of the mean vehicle residency time, $\bar{r}$. It may be seen that the average number of interruptions decreases as the residency time of the vehicles increases.

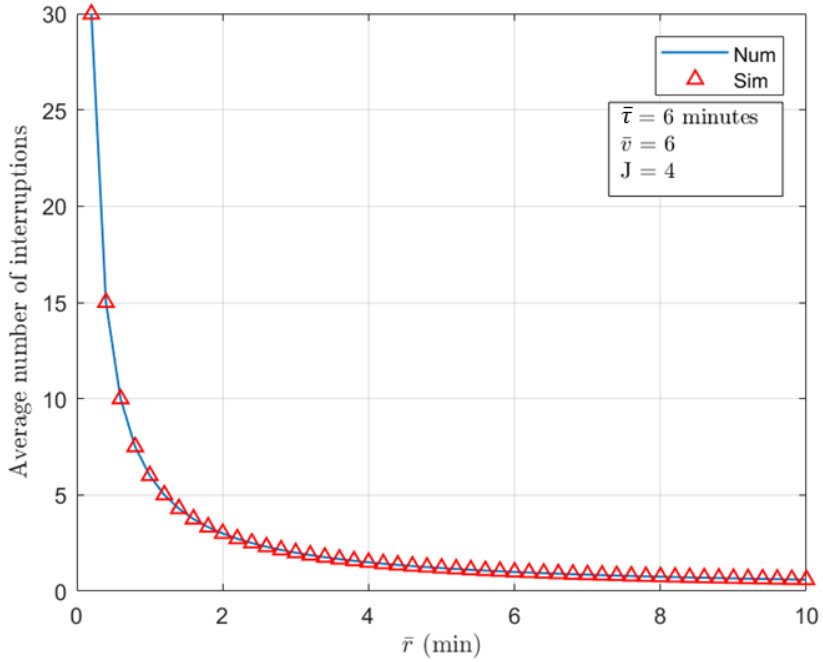


Fig. 2.14. Numerical and simulation results for the average number of interruptions as a function of average vehicle residency time.

2.1.5. Approximate Analysis of Average Job Completion Time with Migration Overhead

Next, we will present an adhoc method to determine an upper bound for average job completion time with migration overhead. Execution of a task will be interrupted each time the worker vehicle serving that task leaves the VC. As a result, the worker vehicle will take time to upload its progress to the leader before it leaves. Similarly, an idle vehicle needs time to download a task from the leader to resume its execution. We let m denote the upload/download time of the VM. Thus, the total migration time overhead of each service interruption will be $2m$. We will assume, in our case, the average size of a VM to be 2 Mbytes. Then given the average data transmission rate is 1 Mbps in 5G NR V2X at 500 m [48], the average migration time, including the overhead, would be approximately 20 seconds.

Simulations have shown that the impact of the migration overhead on the average job completion time is more than the sum of the migration overheads during the job execution time. Let $\bar{Y}_{\mathfrak{J},M}$ denote the average job completion time that includes migration. From the simulation results, we found that it may be estimated as follows,

$$\bar{Y}_{\mathfrak{J},M} = b\left(2m\bar{\ell} + \bar{Y}_{\mathfrak{J}}\right)$$

Where $\bar{\ell}$ is the average number of interruptions from (2.54) and $\bar{Y}_{\mathfrak{J}}$ is the average job completion time without migration overhead, and b is a bias factor introduced to take into account the extra migration overhead. Simulations have shown that the bias factor depends on the average number of vehicles in the VC, assuming that mean residency time is constant. An appropriate choice of the bias factor enabled us to determine an upper bound for average job completion time. In Fig. 2.15 to Fig. 2.17, we present the average job completion times with migration overhead for scenarios with different $\bar{v}$ and a chosen migration time of 10, 20, or 30 seconds. From the comparison with the simulation results, the chosen bias factors result in tight upper bounds for average job completion times.

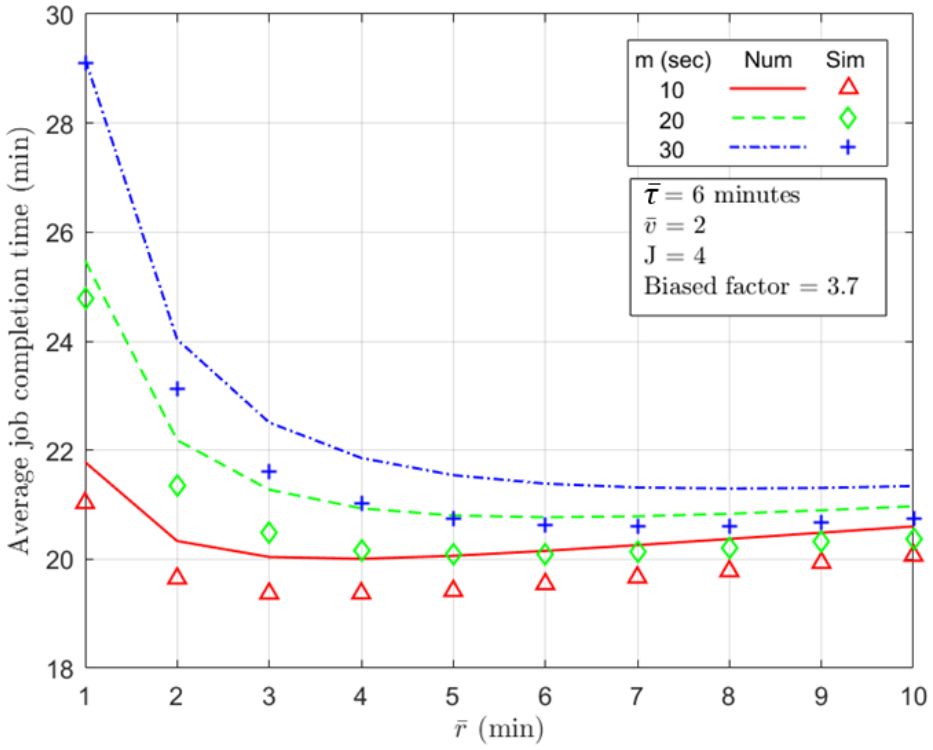


Fig. 2.15. Average job completion time as a function of the average residency time of the vehicles, $\bar{r}$, for case 1 in Table 2.2.

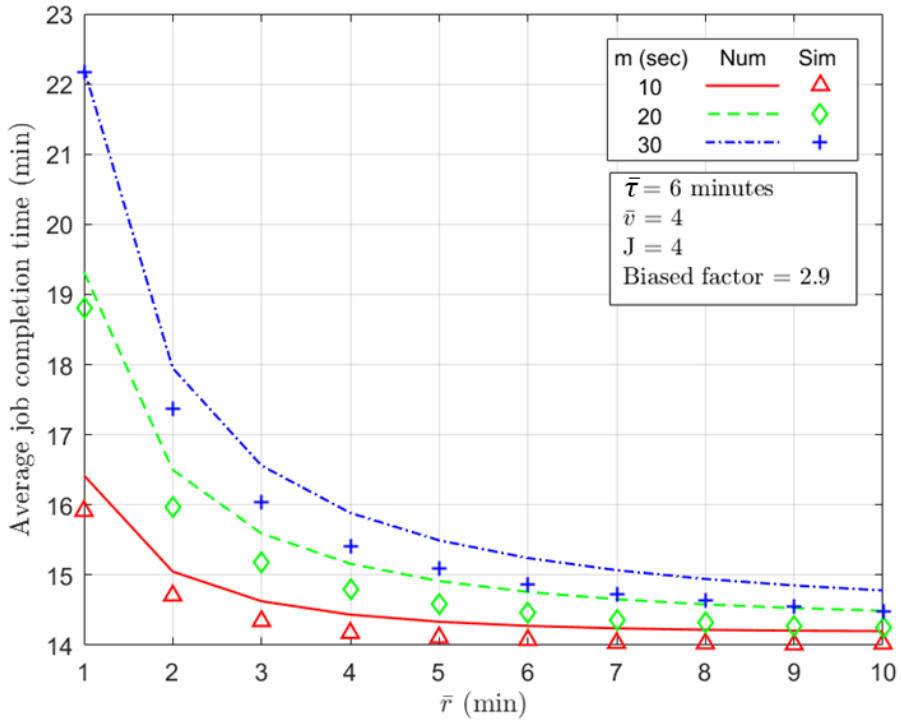


Fig. 2.16. Average job completion time as a function of the average residency times of the vehicles, $\bar{r}$, for case 1 in Table 2.2 and $\bar{v} = 4$.

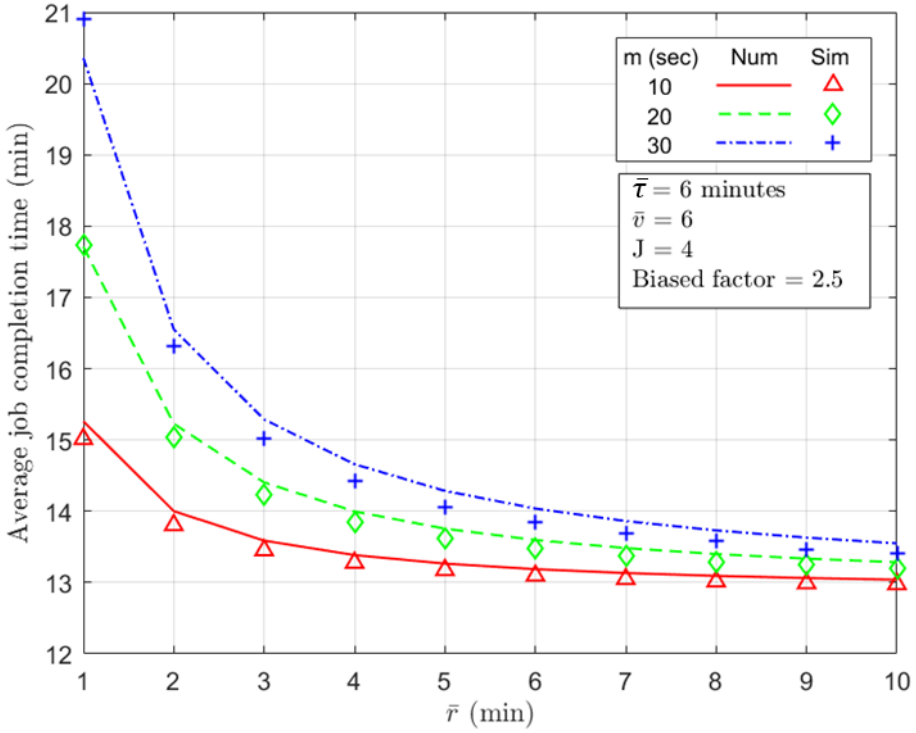


Fig. 2.17. Average job completion time as a function of the average residency times of the vehicles, $\bar{r}$, for case 1 in Table 2.2 and $\bar{v} = 6$.

2.2. Congested traffic on a highway

As previously noted, while highways are often considered uninterrupted flow facilities in traffic engineering, congestion is often seen in the cities. Thus, the vehicle model must be adjusted to reflect the difference between free-flow and congested traffic flow. More specifically, as vehicles slow down when more vehicles are on the highway during the congestion period, their time spent in a VC should increase accordingly.

Regarding the system model, the vehicular cloud model, job model, and task service strategy will remain the same as the work in the previous section. The traffic model is the only difference between this section and the previous section. Therefore, only the important steps of the modeling and analysis will be presented in this section. The details of the analysis may be found in Appendix C.

2.2.1. Modeling approach

In free-flow traffic, the residency times of the vehicles on the highway are independent of each other. As a result, the motion of the vehicles in free-flow traffic is modeled as a $M/M/\infty$ queueing system. In a $M/M/\infty$ system, there is no customer waiting time, and the customer delay in the system equals the customer service time. Under congested traffic, the residency times of the vehicles on the highway are correlated with each other. As a result, the motion of the vehicles in congested traffic is modeled with the $M/M/m$ queuing system, where m is the number of lanes on the highway. In a $M/M/m$ system, customers experience waiting time, and customer delay is given by the sum of its waiting and service times. Since the customers' waiting times depend on each other, customer delays will be correlated. In our application, the customer delay in the queueing system corresponds to the residency time of a vehicle on a highway. Since customer delays in a $M/M/m$ queueing system depend on each other, this will capture the correlation between the residency times of the vehicles under congested traffic. We can find a similar assumption made under a single-lane scenario with a general service time distribution in [122].

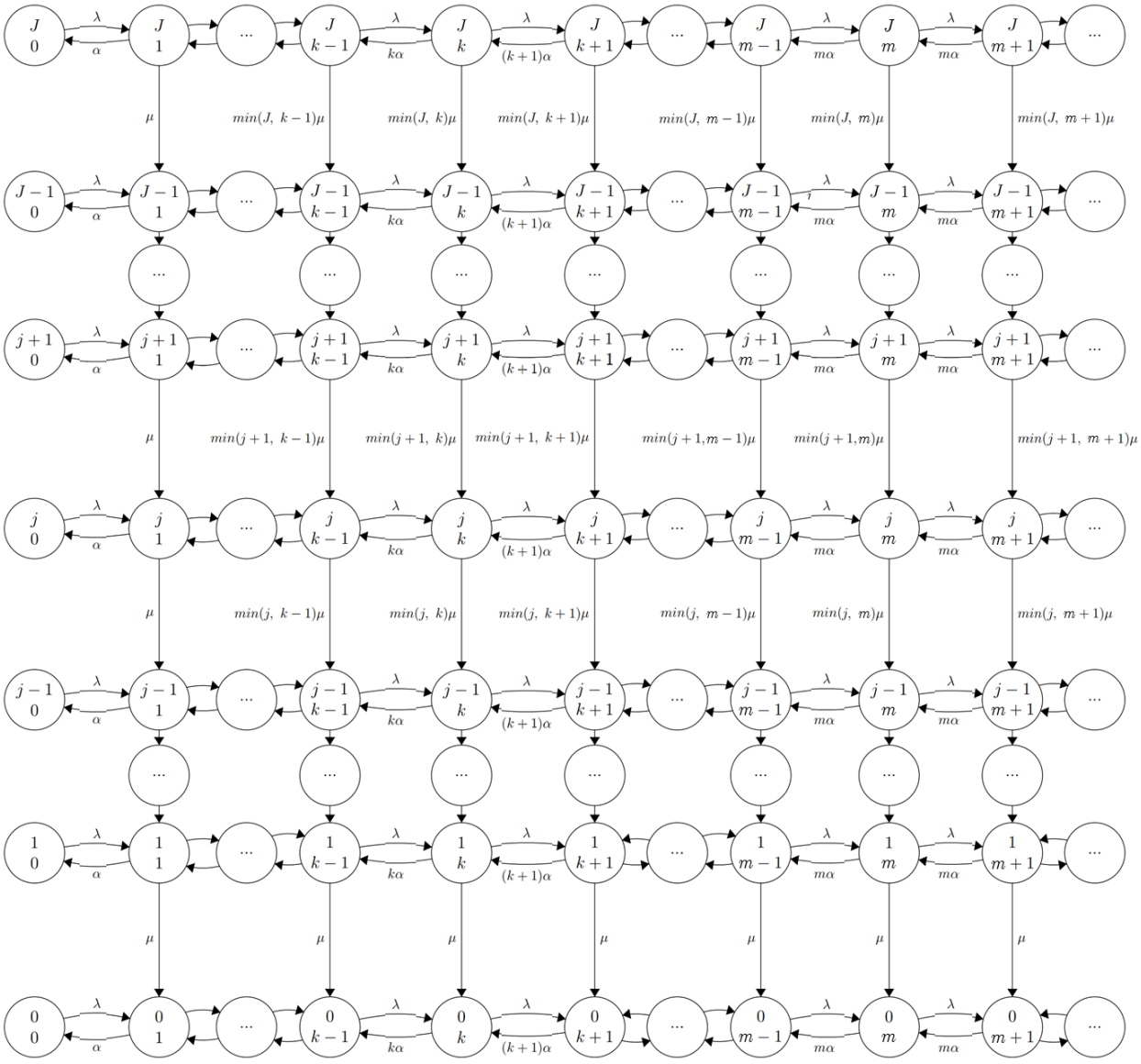


Fig. 2.18. State transition diagram for the system in congested traffic. In each state number of uncompleted tasks is shown above the number of worker vehicles in the system.

Similar to the analysis of the job completion time during a free-flow traffic scenario, a set of two variables, $\{j, k\}$, will represent the state of the system, where j denotes the number of uncompleted tasks, and k denotes the number of worker vehicles in the system at any time. Then we can draw the state transition diagram for the system, as shown in Fig. 2.18. In the figure, in

the states of each row the number of unfinished tasks in the system is same, and in the states of each column the number of vehicles in the VC is same, similar to the case of free-flowing traffic in Fig. 2.3.

The main difference between this model and the previous one is the transitions between the states when there are at least m vehicles in the system. Since the traffic is congested, and the service rate of all the m lanes is same, the rate of vehicles leaving the VC is limited by the number of lanes, m, in the system. More specifically, the rate of vehicles leaving the VC should be $m\alpha$ when there are at least m vehicles in the system, as shown in the columns with at least m vehicles in the system in Fig. 2.18. Additionally, the average wait time of a vehicle in a VC may be nonzero. We note further that the vehicles can serve tasks during their wait time in the VC. Therefore, the vehicles would stay longer in the VC to serve tasks under congested traffic compared to the free-flowing traffic scenario with the same service time. As a result, jobs should complete faster under congested traffic than free-flow traffic when all system parameters are equal.

Then similarly to the decomposition approach in the previous section, we split the system in Fig. 2.18 into subsystems, θ_j, based on the number of remaining uncompleted tasks in the system, j. The state transition of each sub-system is shown in Fig. 2.19.

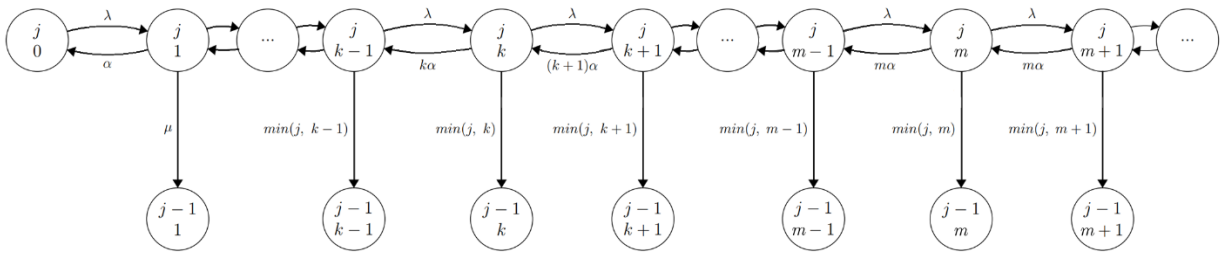


Fig. 2.19. State transition diagram for subsystem θ_j. States $\{j-1, k\}$ are absorption states. Once the transition to states $\{j-1, k\}$ occur, the reverse transition is not allowed.

Due to the similarities of the models between the congested and free-flow scenarios, the analysis is repetitive. As a result, the analysis of the model shown in Fig. 2.19 is presented in Appendix C.

2.2.2. Numerical results

In this section, we present the numerical results of the analysis and Monte Carlo simulation results to verify the accuracy of the analysis.

We obtain the numerical results by solving for $\mathcal{L}_{0,k}(s)$ in (C.26). The operation will require truncation of the infinite summation. In this truncation, we set the upper limit of the sum to be at least twice the mean of the average number of vehicles in the VC. As will be seen, the simulation results validate this choice of the truncation limit. A Monte Carlo simulation program has been written in Matlab, and the results are obtained over 10^6 runs.

Table 2.5 shows the numerical and simulation results of average job completion time, $\bar{Y}_{\mathfrak{J}}$, for different values of average vehicle residency times, $\bar{r}$. We assumed $\mathfrak{J} = 6$ tasks in a job, $m = 4$ lanes on the highway, mean task execution time ($1/\mu = 6\,min$), and mean service time of the vehicles ($1/\alpha = 6\,min$). The results were obtained for different values of vehicle arrival rate. As the vehicle arrival rate increases, vehicle residency times increase because of increasing congestion on the highway. As residency time increases, job completion times decrease since there will be more vehicles in the VC to execute tasks. It may be seen that numerical results match simulation results, showing that the analysis is correct.

$\bar{r}(min)$	7	8	9	10	11	12	13	14	15	16	17	18
ρ	0.589	0.69	0.747	0.785	0.813	0.834	0.851	0.865	0.876	0.886	0.894	0.901
$\bar{Y}_{\mathfrak{J}}$ (Num)	22.05	19.59	18.46	17.77	17.31	16.97	16.71	16.51	16.35	16.21	16.09	15.99
$\bar{Y}_{\mathfrak{J}}$ (Sim)	22.05	19.58	18.44	17.76	17.31	16.97	16.71	16.51	16.35	16.21	16.10	15.99

Table 2.5. Job completion time as a function of average residency time, $\bar{r}$

2.3. Summary

In this chapter, we analyzed the performance of a dynamic VC serving a job with multiple tasks under both free-flow and congested traffic flow on a highway. In free-flow traffic, vehicle residency times are independent of each other, while in congestion, they are correlated. As a result, under free-flow traffic, service to vehicles has been modeled as a $M/M/\infty$, and

under congestion, it has been modeled as a $M/M/m$ queueing system. Since customer waiting times in a $M/M/m$ system are dependent, that takes care of the correlation in the residency times of the vehicles in the congested traffic. In the analysis, we assume that a task can be served by a single vehicle at any time, and a task is not suspended if there are idle vehicles in the VC. A transient analysis is carried out to determine the pdf of the job completion time under zero migration overhead. This analysis is far more difficult than the steady-state analysis of Markov chains. This result provides the minimum job completion time for any task-serving strategy because a task will be served as long as there are idle vehicles in the VC. We show that as the number of vehicles in the system increases, the task suspension times drop and the task service time approaches to the task execution time. We also determine the distribution of the number of service interruptions during task execution, which allows us to infer job completion time under migration overhead. We provide extensive simulation and numerical results to show the accuracy of our analytical model.

.

Chapter 3

Job completion time in a VC with service interruption avoidance strategy

In the service strategy studied in the previous chapter, the execution of a task may be interrupted if the vehicle serving the task leaves the VC. This service strategy may result in excessive virtual machine migration overhead, and further, certain job models may require their tasks to be entirely executed without interruption. In this chapter, we will consider a service strategy that avoids interruption of task execution. In this strategy, a task will only be assigned to a vehicle if the task execution time can be fitted entirely into the residency time of the vehicle. This service strategy directly alleviates the challenges by ensuring that the tasks are executed entirely without being interrupted by migration. However, we can intuitively see that the job completion time under this service strategy may significantly increase depending on the durations of task execution times and vehicle residency times. For instance, if a vehicle does not reside in the VC long enough to complete the task in one go, the task must wait for future vehicles with suitable residency times to arrive at the VC.

In this chapter, for this service strategy, we will derive the distribution of the required number of vehicle arrivals at the VC for the assignment of all the tasks, the bounds of job completion time, and an approximation to job completion time.

3.1. System model

Next, we will describe the system model consisting of vehicular cloud, job model, and task service strategy.

3.1.1. Vehicular cloud and job model

The vehicular cloud and job models are the same as the work in the previous chapter: the VC is formed on a highway by the VC leader, which will recruit nearby moving worker vehicles

to assign the tasks of the job. Upon joining the VC, the workers will let the leader know their routes and speeds, and, based on this information, the leader vehicle will be able to estimate their residency times in the VC.

3.1.2. Task service strategy

A job is assumed to consist of $\mathfrak{J}$ independent tasks in our model, and each task requires a random amount of execution time to be completed. A job will be completed when all its tasks complete their executions. The leader will attempt to assign the vehicles to the remaining unassigned tasks as vehicles arrive. For each arriving vehicle, the leader will assign the longest unassigned task the vehicle can complete within its residency time. In making this assignment decision, the leader will use its estimate of the residency time of the worker vehicle. Finally, the vehicle will return the task execution results to the leader upon task completion and will not be assigned any additional task. In this service strategy, tasks will not experience interruptions in execution and, therefore, no interruptions-related workload migration overhead. Furthermore, a vehicle joining the VC will either execute a single task or remain unassigned during its residency time at the VC. Moreover, the unassigned task with the longest execution time will have the highest priority for assignment to a vehicle. Thus, the first vehicle whose residency time is longer than the longest task execution time will always be the one that will be assigned that task. Another implication of this task service strategy is that the completion time of the last assigned task will not always be the completion time of the job since the last assigned task is not always the longest task. Finally, in this service strategy, a task with a long execution time may substantially increase job completion time since assigning that task to a vehicle may take a long time.

To demonstrate this task service strategy, we give an example in Fig. 3.1 showing how a job of 4 tasks is assigned according to the vehicles' arrival times and residency times in the VC. Firstly, the execution times of the four tasks of a job and their ordered execution times are shown by the bar charts on the left of Fig. 3.1. The residency times of the vehicles arriving at the VC are shown in the bar charts at the upper right of the figure, and the timeline showing the respective arrival time of the vehicles and the completion times of the tasks are at the right bottom of the figure. With respect to the task assignment order, when the first vehicle arrives with the residency time $R_1 = 8\,mins$, it can execute tasks 2, 3 and 4. Therefore, the VC will assign it the

task with the longest execution time, which is task 3 with $X_3 = 7$ mins. This task assignment ensures that the task execution is completed within the residency time of vehicle 1. Next, when vehicle 2 arrives with the residency time $R_2 = 1\ min$, the vehicle cannot execute any remaining tasks (1, 2 and 4) to completion during its residency time, it's not assigned any task. For all the subsequent arrivals, the same logic applies. Thus, upon the arrival of the 5th vehicle, the last task, which is task 2, is assigned. As mentioned before, it is important to note that the last assigned task is not always the task that completes last. As shown in the task assignment timeline in Fig. 3.1, the last task to complete in the job is task 1, which is assigned before tasks 2 and 4. However, since task 1 has a much longer execution time than tasks 2 and 4, its completion time is last. Furthermore, the completion time of task 1 is also the completion time of the job.

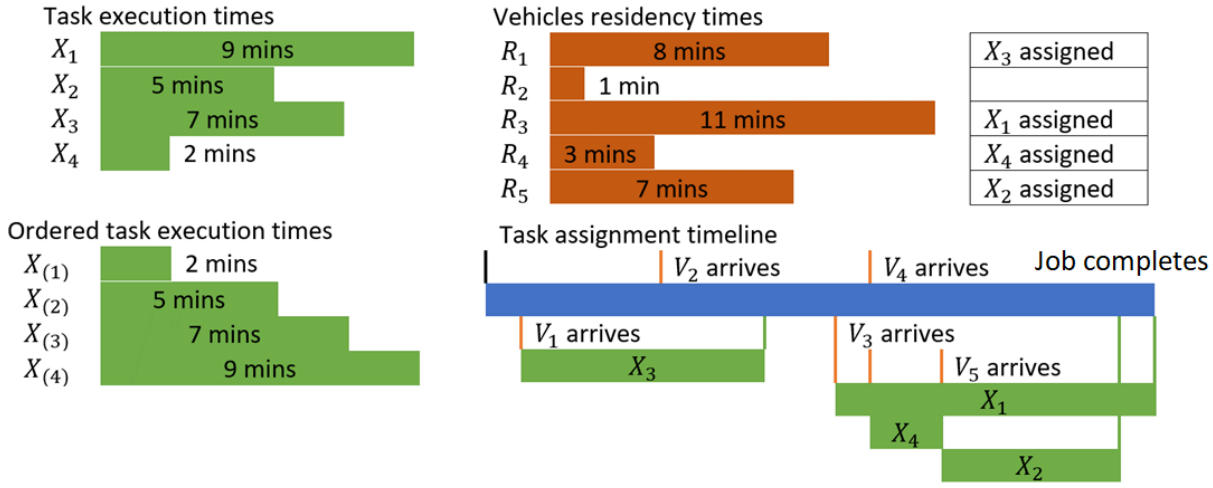


Fig. 3.1. An example of task service strategy for a job with four tasks.

3.1.3. System model assumptions

A job has $\mathfrak{J}$ independent tasks whose execution times are i.i.d random variables with mean $1/\mu$. We assume that vehicles arrive at the VC randomly with mean interarrival time of $1/\lambda$. We further assume that the residency times of vehicles are i.i.d random variables with mean $1/\alpha$. Finally, we assume there are no vehicles in the VC when it is formed.

The main notations employed in this chapter is in Table 3.1.

Notation	Description	Notation	Description
$\mathfrak{J}$	Number of tasks in a job	$g_{R_k}(r_k)$	Probability density function of R_k
K	Number of vehicles have joined the VC	$F_{X_j}(x_j)$	Cumulative distribution function of X_j
$\bar{r}$	Average residency time of vehicles	$G_{R_k}(r_k)$	Cumulative distribution function of R_k
$\bar{s}$	Average execution time of tasks	$f_{X_{(1)},\dots,X_{(\mathfrak{J})}}(x_1, \dots$	Joint probability density function of all order statistics of execution times
$\bar{\tau}$	Average interarrival time of vehicles to the VC	$g_{R_{(1)},\dots,R_{(K)}}(r_1, \dots$	Joint probability density function of all order statistics of vehicle residency times
μ	Service rate of a task	A_K	Event of all tasks are assigned when there have been K vehicles joining a VC
λ	Arrival rate of worker vehicles	$K_{\mathfrak{J}}$	Number of vehicles arrivals at the VC to assign the last task
α	Departure rate of vehicles from the VC	$\mathcal{D}_U$	Upper bound of job completion time
ξ	Upper bound of the execution time of a truncated task	$\mathcal{D}_L$	Lower bound of job completion time
X_j	Random variable as execution time of task j	Y_k	Arrival time of the k^{th} vehicle
R_k	Random variable as residency time of vehicle k	K^*	Number of vehicle arrivals until the longest task is assigned
$X_{(j)}$	j^{th} order statistics of execution times	$Z_{X_{(j)}}$	Completion time of the j^{th} shortest task
$R_{(k)}$	k^{th} order statistics of vehicle residency times	$P_{K_{\mathfrak{J}}}(k)$	Probability mass function of $K_{\mathfrak{J}}$
$f_{X_j}(x_j)$	Probability density function of X_j	$g_{R_k}(r_k)$	Probability density function of R_k
$\mathfrak{J}$	Number of tasks in a job	$F_{X_j}(x_j)$	Cumulative distribution function of X_j
K	Number of vehicles have joined the VC	$G_{R_k}(r_k)$	Cumulative distribution function of R_k
$\bar{r}$	Average residency time of vehicles	$f_{X_{(1)},\dots,X_{(\mathfrak{J})}}(x_1, \dots$	Joint probability density function of all order statistics of execution times
$\bar{s}$	Average execution time of tasks	$g_{R_{(1)},\dots,R_{(K)}}(r_1, \dots$	Joint probability density function of all order statistics of vehicle residency times
$\bar{\tau}$	Average interarrival time of vehicles to the VC		

Table 3.1. Main notations of this chapter

3.2. Mathematical preliminaries

We assume that task execution times have a general distribution that is supported by a semi-infinite interval of $(0, \infty)$. As noted earlier, a task with a long execution time may significantly increase job completion time. As a result, we will also consider the impact of the bounding of the task execution times on job completion time. We let $f_X(x)$ and $F_X(x)$ denote the pdf and the cdf of task execution times, respectively. We can set upper bound for the task execution times to ξ by truncating the pdf as follows,

$$f_X(x|X \leq \xi) = \frac{f_X(x)}{F_X(\xi)} \tag{3.1}$$

For task execution times that are exponentially distributed random variables with mean $1/\mu$, the truncated pdf from (3.1) becomes,

$$f_X(x|X \leq \xi) = \frac{\mu e^{-\mu x}}{(1 - e^{-\mu \xi})} \tag{3.2}$$

We let X_j denote the execution time of j'th task for $1 \leq j \leq \Im$, and R_k denote the residency time of the k'th vehicle, for $1 \leq k \leq K$, where K is the number of vehicle arrivals at the VC. We assume that X_j and R_k are i.i.d for $1 \leq j \leq \Im$ and $1 \leq k \leq K$. We further denote the cdfs and pdfs of X_j and R_k as $F_X(x)$, $f_X(x)$ and $G_R(r)$, $g_R(r)$, respectively. Let us arrange all X_j and all R_k among themselves according to the order of their magnitudes as follows,

$$X_{(1)} \leq \ldots \leq X_{(j)} \leq \ldots \leq X_{(\Im)}, \qquad R_{(1)} \leq \ldots \leq R_{(k)} \leq \ldots \leq R_{(K)} \tag{3.3}$$

then $X_{(j)}$, $R_{(k)}$ are known as the j^{th} and k^{th} order statistics, respectively. From [39] the pdfs of $X_{(j)}$ and $R_{(k)}$ are given by,

$$f_{X_{(j)}}(x) = \frac{\Im!}{(j-1)!\,(\Im - j)!} f_X(x)[F_X(x)]^{j-1}[1 - F_X(x)]^{\Im - j} \tag{3.4}$$

$$g_{R_{(k)}}(r) = \frac{K!}{(k-1)!\,(K-k)!} g_R(r)[G_R(r)]^{k-1}[1 - G_R(r)]^{K-k} \tag{3.5}$$

We let $f_{X_{(1)},\dots,X_{(\Im)}}(x_1,\dots,x_\Im)$ and $g_{R_{(1)},\dots,R_{(K)}}(r_1,\dots,r_K)$ denote the joint distributions of all $X_{(j)}$ and all $R_{(k)}$, respectively, then

$$f_{X_{(1)},\dots,X_{(\Im)}}(x_1,\dots,x_\Im) = \Im! \prod_{i=1}^{\Im} f_{X_i}(x_i) \tag{3.6}$$

$$g_{R_{(1)},\dots,R_{(K)}}(r_1,\dots,r_K) = K! \prod_{i=1}^{K} g_{R_i}(r_i) \tag{3.7}$$

3.3. Analysis of job completion time

In this section, the probability distribution of the number of vehicle arrivals to assign all tasks of a job, the probability density function of the upper and lower bounds of the job completion time, and the probability density function of the completion time of the longest task will be determined.

3.3.1. Probability distribution of the number of vehicle arrivals for the assignment of all the tasks in a job

In this subsection, we will determine the probability distribution of the required number of vehicle arrivals to assign all the tasks in a job. For all the tasks to be assigned, there must have been at least $\Im$ vehicle arrivals at the VC. Let A_K denote the event that all the tasks have been assigned when there are K vehicle arrivals to the VC. If $K \geq \Im$, it means that $R_{(K-\Im+i)} > X_{(i)}, \forall i = 1..\Im$. The probability of the event A_K is given by,

$$Pr(A_K) = \begin{cases} Prob\left(R_{(K)} > X_{(\Im)}, \dots, R_{(K-\Im+1)} > X_{(1)}\right), & K \geq \Im \\ 0, & K < \Im \end{cases} \tag{3.8}$$

We note that the probability in (3.8) only involves the highest $\Im$ residency times, whose joint pdf can be determined from (3.7) as,

$$g_{R_{(K)},\dots,R_{(K-\Im+1)}}(r_K,\dots,r_{K-\Im+1}) \tag{3.9}$$

$$= \int_0^{r_{K-\Im+1}} \dots \int_0^{r_2} g_{R_{(K)},\dots,R_{(1)}}(r_K, \dots, r_1) dr_1 \dots dr_{K-\Im}$$

If the residency times of the vehicles are exponentially distributed, then integrations in (3.9) can be evaluated, which results in the following closed form (derivation of this result is given in Appendix D),

$$\begin{aligned} & g_{R_{(K)},\dots,R_{(K-\Im+1)}}(r_K, \dots, r_{K-\Im+1}) \\ & = K!\, \alpha^{\Im} e^{-\alpha(r_K+\dots+r_{K-\Im+1})} \cdot \frac{e^{-(K-J)\alpha r_{K-J+1}} (e^{\alpha r_{K-J+1}} - 1)^{K-J}}{(K-J)!} \end{aligned} \tag{3.10}$$

To determine the probability in (3.8), we need to derive the joint pdf of the ordered vehicle residency times and the ordered task execution times, which is denoted as $h_{R_{(K)},\dots,R_{(K-\Im+1)},X_{(\Im)},\dots,X_{(1)}}(r_K, \dots, r_{K-\Im+1}, x_\Im, \dots x_1)$. Since task execution times are independent of vehicle residency times, the joint pdf is given by,

$$\begin{aligned} & h_{R_{(K)},\dots,R_{(K-\Im+1)},X_{(\Im)},\dots,X_{(1)}}(r_K, \dots, r_{K-\Im+1}, x_\Im, \dots x_1) \\ & \qquad = g_{R_{(K)},\dots,R_{(K-\Im+1)}}(r_K, \dots, r_{K-\Im+1}) f_{X_{(\Im)},\dots,X_{(1)}}(x_\Im, \dots, x_1) \end{aligned} \tag{3.11}$$

Then, $P(A_k)$ can be determined as,

$$Pr(A_K) = \int \dots \int_D h_{R_{(K)},\dots,X_{(1)}}(r_K, \dots, x_1)\, dx_1 \dots dr_K \tag{3.12}$$

where in the above D is the domain of the integration. Let $D_{untruncated}$ and $D_{truncated}$ denote the domain of integration for untrancated and truncated execution times respectively. Then, they are given by (3.13) and (3.14),

$$\begin{aligned} D_{untruncated} = \{ & (x_1, \dots, x_\Im, r_{K-\Im+1}, \dots, r_K) : (x_i \le r_j \forall j = K-\Im+i, 1 \le i \\ & \le \Im - 1)\ and\ (x_i \le x_{i+1} \forall 1 \le i \le \Im - 1)\ and\ (r_j \le r_{j+1} \forall K-\Im+1 \\ & \le j \le K-1)\} \end{aligned} \tag{3.13}$$

$$D_{truncated} = \{(x_1, \dots, x_{\Im}, r_{K-\Im+1}, \dots, r_K): (x_i \le r_j \forall j = K - \Im + i, 1 \le i \le \Im - 1) \; and \; (x_i \le x_{i+1} \forall 1 \le i \le \Im - 1) \; and \; (r_j \le r_{j+1} \forall K - \Im + 1 \le j \le K - 1) \; and \; (x_i \le \xi \forall 1 \le i \le \Im)\} \tag{3.14}$$

As an example, let us assume that we have $\Im = 2$ tasks, then the domains of the integration in (3.13) and for the untruncated and truncated cases are, respectively,

$$D_{untruncated}^{\Im=2} = \{(0 \le x_1 < x_2 < r_{K-1} < r_K < \infty) \vee (0 \le x_1 < r_{K-1} < x_2 < r_K < \infty)\} \tag{3.15}$$

$$\begin{aligned} D_{truncated}^{\Im=2} = \{&(0 \le x_1 < x_2 < \xi < r_{K-1} < r_K < \infty) \\ &\vee (0 \le x_1 < x_2 < r_{K-1} < \xi < r_K < \infty) \\ &\vee (0 \le x_1 < x_2 < r_{K-1} < r_K < \xi) \\ &\vee (0 \le x_1 < r_{K-1} < x_2 < \xi < r_K < \infty) \\ &\vee (0 \le x_1 < r_{K-1} < x_2 < r_K < \xi)\} \end{aligned} \tag{3.16}$$

From (3.15) and (3.16), (3.12) can be rewritten for the untruncated and truncated cases as shown in (3.17) and (3.18), respectively. We first denote $h_{R_{(K)},R_{(K-1)},X_{(2)},X_{(1)}}(r_K, r_{K-1}, x_2, x_1)$ as $\mathbb{H}_2$, then,

$$\begin{aligned} P\left(A_{K,untruncated}^{\Im=2}\right) &= \int_0^{\infty} \int_0^{r_K} \int_0^{r_{K-1}} \int_0^{x_2} \mathbb{H}_2 dx_1 dx_2 dr_{K-1} dr_K \\ &+ \int_0^{\infty} \int_0^{r_K} \int_0^{x_2} \int_0^{r_{K-1}} \mathbb{H}_2 dx_1 dr_{K-1} dx_2 dr_K \end{aligned} \tag{3.17}$$

$$P\left(A_{K,truncated}^{\Im=2}\right) =$$

$$= \int_{\xi}^{\infty}\int_{\xi}^{r_K}\int_{0}^{\xi}\int_{0}^{x_2} \mathbb{H}_2 dx_1 dx_2 dr_{K-1} dr_K$$

$$+ \int_{\xi}^{\infty}\int_{0}^{\xi}\int_{0}^{r_{K-1}}\int_{0}^{x_2} \mathbb{H}_2 dx_1 dx_2 dr_{K-1} dr_K$$

$$+ \int_{0}^{\xi}\int_{0}^{r_K}\int_{0}^{r_{K-1}}\int_{0}^{x_2} \mathbb{H}_2 dx_1 dx_2 dr_{K-1} dr_K$$

$$+ \int_{\xi}^{\infty}\int_{0}^{\xi}\int_{0}^{x_2}\int_{0}^{r_{K-1}} \mathbb{H}_2 dx_1 dr_{K-1} dx_2 dr_K$$

$$+ \int_{0}^{\xi}\int_{0}^{r_K}\int_{0}^{x_2}\int_{0}^{r_{K-1}} \mathbb{H}_2 dx_1 dr_{K-1} dx_2 dr_K$$

(3.18)

Let $K_{\Im}$ denote the number of vehicle arrivals when the last task is assigned, and define $P_{K_{\Im}}(k)$ as the probability distribution of $K_{\Im}$,

$$P_{K_{\Im}}(k) = Pr\left(K_{\Im} = k\right) \tag{3.19}$$

Then, the probability of the event A_K may be expressed in terms of $P_{K_{\Im}}(k)$ as,

$$Pr(A_K) = \sum_{k=\Im}^{K} P_{K_{\Im}}(k) \tag{3.20}$$

From the above, finally, we determine the probability distribution of the number of vehicle arrivals for the assignment of all the tasks in a job,

$$P_{K_{\Im}}(K) = \begin{cases} Pr(A_K) - Pr(A_{K-1}), & K > \Im \\ P(A_K), & K = \Im \\ 0, & K < \Im \end{cases} \tag{3.21}$$

3.3.2. Probability density function of the upper and lower bound of job completion time

Next, we will derive the pdf of the upper and lower bound of job completion time. First, let us determine the pdf of the vehicle's arrival time assigned to the last task. Let Y_k denote the arrival time of the k'th vehicle and $f_{Y_k}(t)$ its pdf. Let τ_k denote the interarrival time of the $k'th$ vehicle, then,

$$Y_k = \sum_{i=1}^{k} \tau_i \tag{3.22}$$

Since $\tau_k \forall k$ is assumed to be i.i.d with pdf $f_\tau(t)$, Laplace transform of Y_k could be obtained as follows. Let us define $Y_k(s) = E[e^{-sY_k}]$ and $\tau(s) = E[e^{-s\tau}]$ as Laplace transforms of $f_{Y_k}(t)$ and $f_\tau(t)$, respectively. Then,

$$Y_k(s) = \prod_{i=1}^{k} \tau_i(s) = [\tau(s)]^k \tag{3.23}$$

If $\tau_k \, \forall k$ are exponentially distributed, then the pdf of Y_k is given by the Erlang distribution [121],

$$f_{Y_k}(t) = \frac{\lambda^k t^{k-1} e^{-\lambda t}}{(k-1)!} \tag{3.24}$$

Let $Y_{K_\Im}$ denote the arrival time of the vehicle which has been assigned to the last task. Since, distribution of the number of vehicle arrivals to assign the last task is given in (3.21), using the law of total probability, we determine the pdf of $Y_{K_\Im}$ as follows,

$$f_{Y_{K_\Im}}(t) = \sum_{k=\Im}^{\infty} f_{Y_k}(t) P_{K_\Im}(k) \tag{3.25}$$

Clearly, the job completion time will be the longest when the longest task is assigned for execution last. Otherwise,job completion time will be the shortest when the shortest task is

assigned to execute last. Let $\mathcal{D}_U$ and $\mathcal{D}_L$ denote the upper and lower bound of the job completion time, then,

$$\mathcal{D}_U = Y_{K_\Im} + X_{(\Im)} \tag{3.26}$$

$$\mathcal{D}_L = Y_{K_\Im} + X_{(1)} \tag{3.27}$$

The pdfs of $\mathcal{D}_U$ and $\mathcal{D}_L$ can be determined from (3.26), (3.27) using convolution property as follows

$$f_{\mathcal{D}_U}(t) = f_{Y_{K_\Im}}(t) * f_{X_{(\Im)}}(t) = \int_0^\infty f_{Y_{K_\Im}}(x) f_{X_{(\Im)}}(t-x)dx \tag{3.28}$$

$$f_{\mathcal{D}_L}(t) = f_{Y_{K_\Im}}(t) * f_{X_{(1)}}(t) = \int_0^\infty f_{Y_{K_\Im}}(x) f_{X_{(1)}}(t-x)dx \tag{3.29}$$

3.3.3. Mean of upper and lower bound of job completion time

Next, we will determine the mean of the job completion time's upper and lower bound. From (3.26) and (3.27),

$$E[\mathcal{D}_U] = E\left[Y_{K_\Im}\right] + E\left[X_{(\Im)}\right] \tag{3.30}$$

$$E[\mathcal{D}_L] = E\left[Y_{K_\Im}\right] + E\left[X_{(1)}\right] \tag{3.31}$$

Similarly to the reasoning for the derivation of (3.25), the expected arrival time of the vehicle which has been assigned to the last task is given by,

$$E\left[Y_{K_\Im}\right] = \sum_{k=\Im}^{\infty} E[Y_k] P_{K_\Im}(k) \tag{3.32}$$

Since the interarrival times are i.i.d., from (3.22), $E[Y_k] = kE[\tau]$, where $E[\tau] = {}^1/_\lambda$. Substituting this in (3.32) and then (3.32) in (3.30) and (3.31) gives the mean of the upper and lower bound of job completion time as

$$E[\mathcal{D}_U] = \lambda E\left[K_{\Im}\right] + E\left[X_{(\Im)}\right] \tag{3.33}$$

$$E[\mathcal{D}_L] = \lambda E\left[K_{\Im}\right] + E\left[X_{(1)}\right] \tag{3.34}$$

3.3.4. Probability density function of the longest task completion time

Due to the service strategy, the completion time of the longest task execution is expected to be a good approximation of the job completion time. As a result, the pdf of the longest task completion time will be derived in this subsection. Since the longest task has the highest assignment priority, when a vehicle's residency time, R, is larger than the execution time of the longest task, $X_{(\Im)}$, it is the only possible task that can get assigned. Let us assume that the longest task execution time is a constant, then the comparisons of R and $X_{(\Im)}$ are independent Bernoulli trials. Let p^* denote the probability that the outcome of a trial is a "success," we then have

$$p^* = Prob\left(R > X_{(\Im)} | X_{(\Im)} = c\right) \tag{3.35}$$

Let K^* denote the number of the vehicle that has been assigned the longest task. Then the random variable K^* has a geometric distribution with parameter p^*:

$$P_{K^*|X_{(\Im)}=c}\left(K^* = k | X_{(\Im)} = c\right) = (1-p^*)^{k-1}p^*, \qquad k \geq 1 \tag{3.36}$$

We further denote the arrival time of the vehicle serving the longest task as Y_{K^*}. Similarly to the derivation of $f_{Y_{K_{\Im}}}(t)$ in equations (3.22) to (3.25), we can determine the pdf of Y_{K^*} as:

$$f_{Y_{K^*}|X_{(\Im)}=c}\left(t | X_{(\Im)} = c\right) = \sum_{k=1}^{\infty} f_{Y_k}(t) P_{K^*|X_{(\Im)}=c}\left(K^* = k | X_{(\Im)} = c\right) \tag{3.37}$$

When the interarrival time of the vehicles is exponentially distributed, substituting in the above from (3.24), we have,

$$f_{Y_{K^*}|X_{(\Im)}=c}\left(t|X_{(\Im)}=c\right)=\sum_{m=1}^{\infty}\frac{\lambda^m t^{m-1}e^{-\lambda t}}{(m-1)!}P_{K^*|X_{(\Im)}=c}\left(K^*=k|X_{(\Im)}=c\right)$$

$$=\lambda(1-p^*)e^{-\lambda t}\sum_{k=1}^{\infty}\frac{(\lambda p^* t)^{k-1}}{(k-1)!}=\lambda(1-p^*)e^{-(1-p^*)\lambda t} \tag{3.38}$$

Let us finally denote the completion time of the longest task as $Z_{X_{(\Im)}}$. Then,

$$Z_{X_{(\Im)}}=Y_{K^*}+X_{(\Im)} \tag{3.39}$$

Assuming that $X_{(\Im)}$ is a constant, the conditional pdf of $Z_{X_{(\Im)}}$ can be written as

$$f_{Z_{X_{(\Im)}}|X_{(\Im)}=c}\left(z\middle|X_{(\Im)}=c\right)=\begin{cases}0, & z\leq c\\ f_{Y_{K^*}|X_{(\Im)}=c}\left(z-c|X_{(\Im)}=c\right), & z>c\end{cases} \tag{3.40}$$

Since the pdf of $X_{(\Im)}$ can be determined from (3.4), we can finally find the unconditional pdf of the longest task completion time as :

$$f_{Z_{X_{(\Im)}}}(z)=\int_0^{\infty}f_{Z_{X_{(\Im)}}|X_{(\Im)}=x_{\Im}}\left(z\middle|X_{(\Im)}=x_{\Im}\right)f_{X_{(\Im)}}\left(x_{\Im}\right)dx_{\Im} \tag{3.41}$$

3.4. Numerical results

In this section, we will present the representative numerical and simulation results to show the correctness of our analysis. It is noted that the interarrival times and the residency times of the vehicles will be chosen to be exponentially distributed in this section. We let $\bar{r}$ denote the average residency time of a vehicle, $\bar{s}$ the average execution time of a task and $\bar{\tau}$ the average vehicle interarrival time. Simulation has been implemented in Matlab, and it has been described in Appendix E.

Fig. 3.2 plots the probability of the event that all the tasks of a job will be assigned before or at *K*'th vehicle, $Pr(A_K)$, as a function of the number of vehicle arrivals with average task execution time as a parameter. Fig. 3.3 plots the probability of the event of assigning the last task of a job at the *k*'th vehicle, $P_{K_{\Im}}(k)$, as a function of the number of vehicle arrivals for the same

system. We first note that the numerical and simulation results agree very well in both figures. Furthermore, the figures support our intuitive understanding of the system's behavior. As the average task execution times increases with constant vehicle residency times, finding a vehicle that can execute a task within the vehicle's residency period becomes harder. Thus, the VC will need more vehicles joining before finding the ones that can complete the job. This reasoning is confirmed in Fig. 3.3 when the $P_{K_{\mathfrak{J}}}(k)$ curve for $\bar{s} = 8\,minutes$, denoted by $P_{K_{\mathfrak{J}}}^{\bar{s}=8}(k)$, is compared with the $P_{K_{\mathfrak{J}}}(k)$ curve for $\bar{s} = 2\,minutes$, denoted $P_{K_{\mathfrak{J}}}^{\bar{s}=2}(k)$, at $k = 4$ and $k = 24$: It may be seen that $P_{K_{\mathfrak{J}}}^{\bar{s}=8}(4) < P_{K_{\mathfrak{J}}}^{\bar{s}=2}(4)$ and $P_{K_{\mathfrak{J}}}^{\bar{s}=8}(24) > P_{K_{\mathfrak{J}}}^{\bar{s}=2}(24)$ which confirms our intuition.

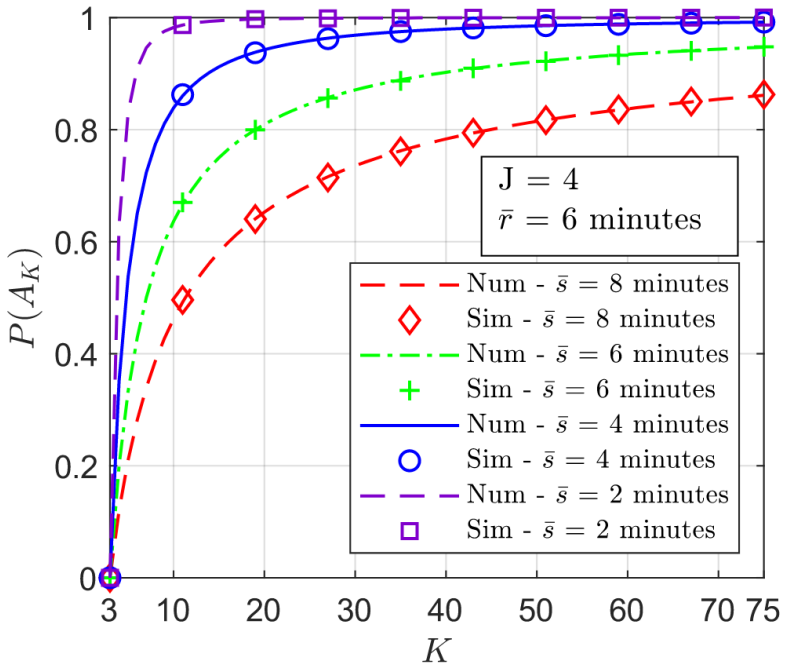


Fig. 3.2. Numerical and simulation results of probability of assigning all the tasks of a job as a function of the number of vehicle arrivals with average task execution time as a parameter and constant average residency time.

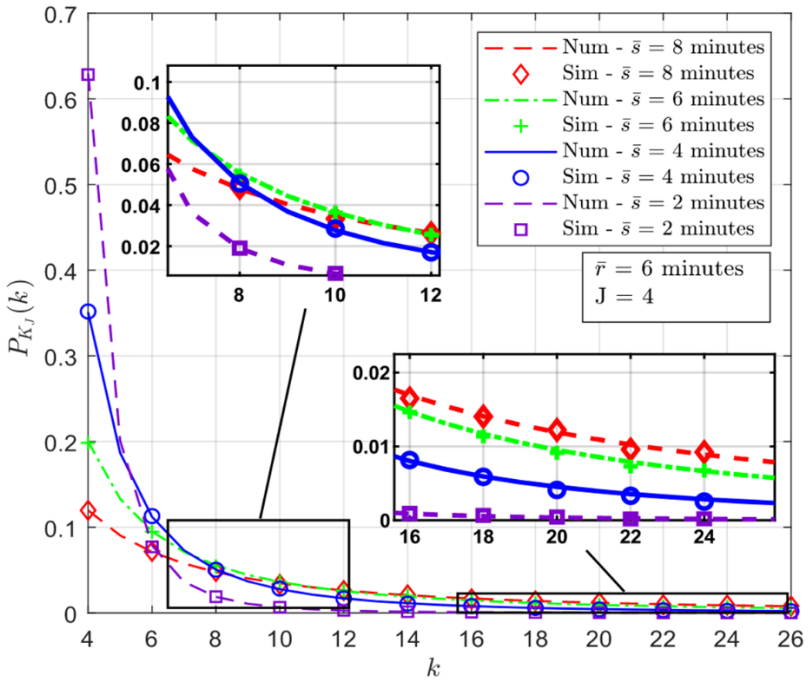


Fig. 3.3. Numerical and simulation results of probability of assigning the last task to the k'th vehicle as a function of k with average task execution time as a parameter.

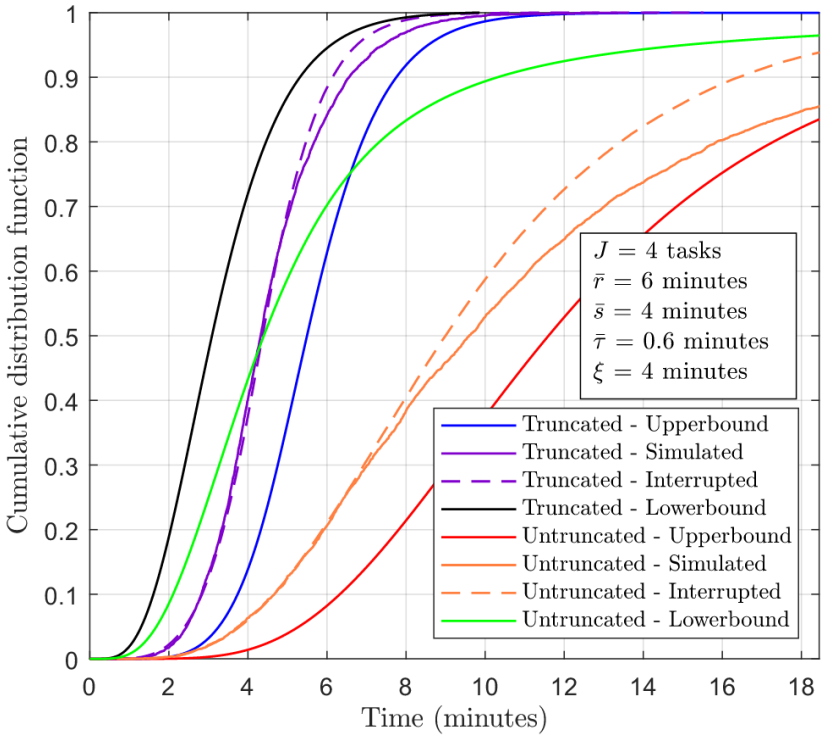


Fig. 3.4. Numerical and simulation results of upper and lower bound of job completion time for truncated and untruncated task execution times.

Fig. 3.4 plots the cdf curves of the upper and lower bounds of job completion times $Prob(D_U < t)$, $Prob(D_L < t)$ and the simulation results of job completion times $Prob(D < t)$ under uninterrupted and interrupted task assignment schemes. Further, results have been plotted both for truncated and untruncated task execution times. In the truncated cases, the upperbound for task execution time is chosen to be $\xi = 4\,min$. It may be seen that simulation results for job completion time fall between the upper and lower bounds for both truncated and untruncated task execution times. It's also seen that truncation of the distribution of the task execution time effectively reduces the mean task execution time and job completion time and bounds. Finally, it's seen that the system's performance under interrupted service with no migration overhead is better than uninterrupted service. This observation is an example of the previous conclusion that the pdf of job completion time under interrupted task assignment serves as the lowest bound of any task servicing discipline.

Fig. 3.5 plots the numerical and simulation results of the pdf of the longest task completion time and the simulation result of job completion time for untruncated task execution times. Again, it may be seen that numerical and simulation results for the longest task completion time agree well. Further, the results show that the completion time of the longest task is a good approximation for the job completion time for this example.

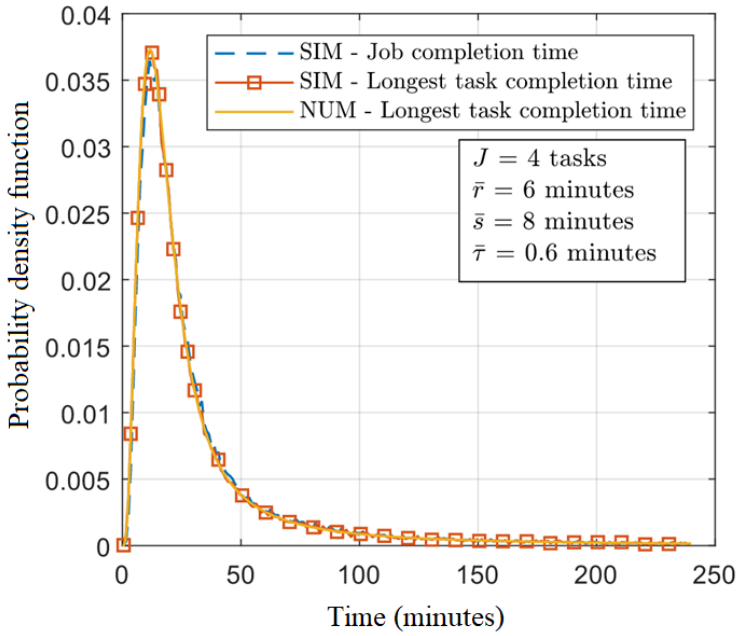


Fig. 3.5. Numerical and simulation results for the pdf of the completion time of the longest task for untruncated task execution time.

Fig. 3.6 presents the numerical and simulation results of the average completion time of the longest task and the average simulation job completion time as a function of the number of tasks in the job for truncated task execution times with bound $\xi = 8$ and $\xi = 18$ mins. It may be seen that as the number of tasks in a job increases, the completion time of the longest task will no longer be a good approximation for the completion time of the job. This may be due to having other tasks with execution times close to the task with the longest execution time. Despite this difference, as ξ increases, the difference between these two vanishes regardless of the number of tasks in a job, an example of which is shown in Fig. 3.7. This means that for the untruncated case, the average job completion time can be approximated very well by the average longest task completion time.

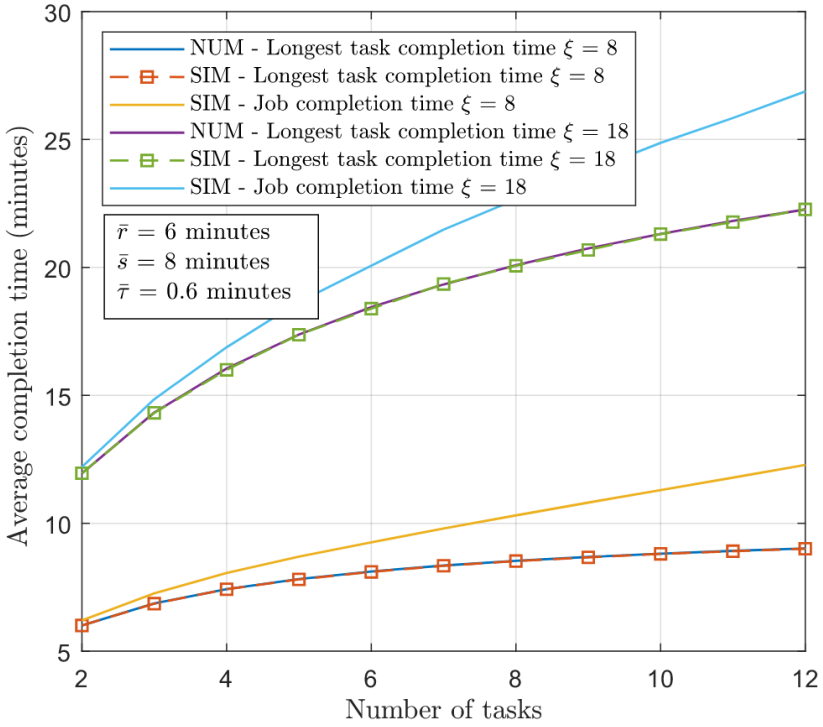


Fig. 3.6. Numerical and simulation results for the mean completion time of the task with longest truncated execution time as a function of the number of tasks in a job.

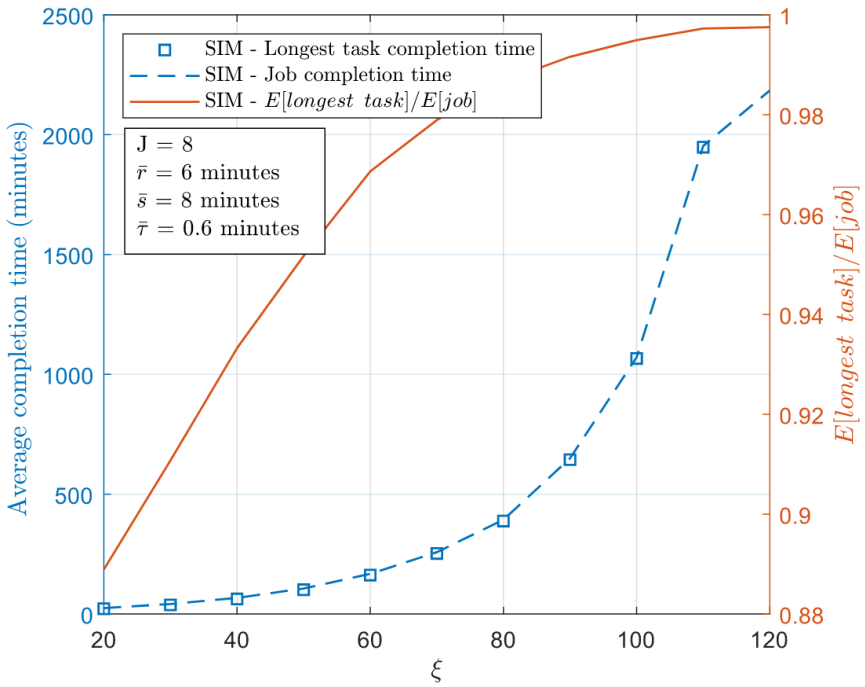


Fig. 3.7. Simulation results for average completion time of the longest task and the job and their ratio as a function of task execution bound.

3.5. Summary

In the VC architecture, there may be cases that task migration may not be desirable because of its overhead or due to the Quality of Service (QoS) requirements of the application. In these cases, the VC has to employ a service strategy to avoid workload migration caused by interruptions. This chapter employs the strategy that the leader only assigns a task to a vehicle when the vehicle can finish the task during its residency time in the VC. The arrival times, the vehicles' residency times, and the execution times of the tasks in a job are assumed to be i.i.d with arbitrary probability distributions. Based on this assumption and the interruption-avoidance service strategy, we provide the probability distribution of the number of vehicles arriving at the VC to assign all the tasks in a job. Based on this foundational result, the probability density function of the bounds of the job completion times and their first moments are derived. We derive the pdf of the competion time of the task with longest execution time. We show that average longest task's completion time is a good approximation for the average job completion time. The numerical results show that if a job has tasks with long execution times this strategy will result in long job completion times. As a result, it will be good to use a hiybrid strategy,

where short tasks are assigned according to the interruption avoidance strategy and long tasks according to the strategy that allows interruption. Finally, Monte Carlo simulation results are used to verify the numerical results of the analysis.

Chapter 4

Computing capacity of a VC with service interruption avoidance strategy

In this chapter, we will derive the computing capacity of a VC during its lifetime for service interruption avoidance strategy. The computing capacity gives us the distribution of the number of jobs a VC can execute during its existence.

4.1. System model

In this section, we describe vehicular cloud, job, and service models and state the mathematical assumptions.

4.1.1. Vehicular Cloud model

We assume a highway with free-flow traffic where vehicles can choose to interconnect and form a VC through wireless communications. The vehicle that initiates the formation of the VC becomes its leader. The leader recruits other vehicles and manages and assigns computing tasks to the vehicles in the VC. A vehicle joining the VC agrees the VC to utilize its computing resources and must connect directly to the leader to join the VC. We assume that the connectivity range of the leader vehicle is sufficiently large. A vehicle may leave the VC when it exits from the highway or if it decides to stop sharing its resources. The period that a vehicle is a member of a VC will be referred to as its residency time. A vehicle joining to VC will share its path and speed with the VC leader. Then, the leader will be able to estimate the residency times of the joining vehicles. When the leader leaves the VC, its role is transferred to another member while guaranteeing connectivity to the remaining members. The lifetime of a VC ends when the leader is the only vehicle in the VC.

4.1.2. Job and service model

A job contains a number of tasks, which can either come from the VC members or external mobile devices connected to the VC called a requester. A job may have random number of tasks with Poisson distribution with parameter γ. The execution times of the tasks will be i.i.d. random variables with exponential distribution with parameter μ. We assume that the VC continuously receives jobs until the last vehicle leaves the VC; therefore, the VC never runs out of jobs to execute.

In this work, a vehicle will keep executing tasks one after another during its residency time in the VC. However, we assume an uninterrupted task service strategy which means that a vehicle will only be assigned a task if it can finish its execution before leaving the VC. An example of the job and service models are given in Fig. 4.1. In part a of the figure, we have job 1 and 2 with 5 and 4 tasks in each job, respectively. These tasks are ready to be assigned at the conception of the VC. In part b of the figure, it shows that each vehicle executes tasks subsequently, and the vehicles are only assigned the next task only if it finishes the current task. This approach continues until the remaining residency times of the vehicles no longer sufficiently long to execute the task entirely. Then the vehicle will be idle.

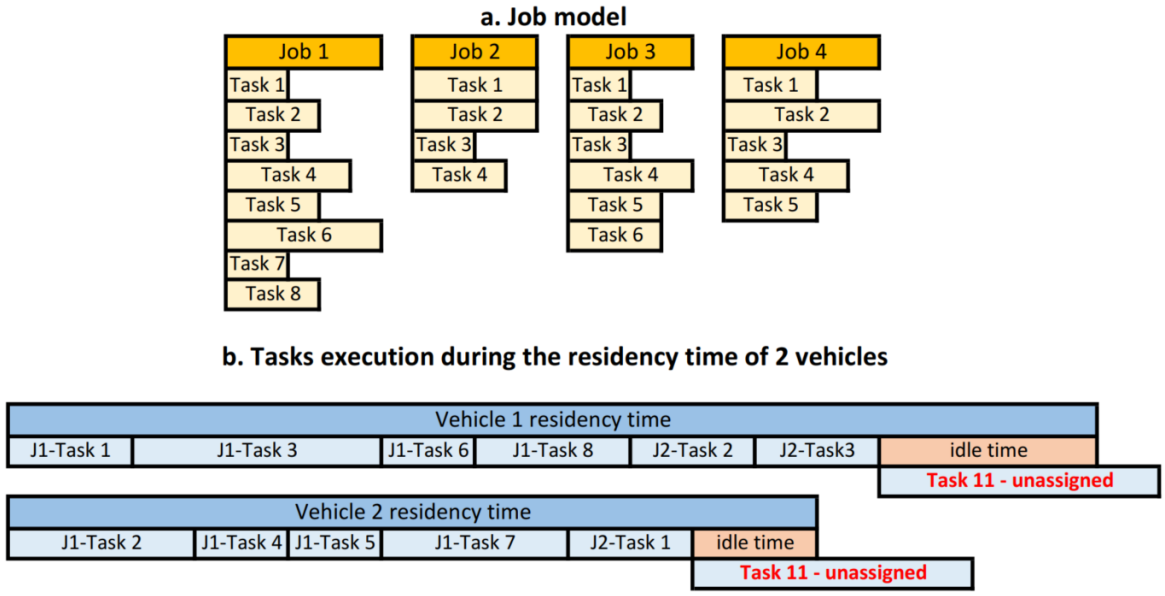


Fig. 4.1. An example of the job model and service strategy.

Notation	Description	Notation	Description
μ	Service rate of a task	J_i	Number of tasks served by the i'th vehicle, $i = 1 \dots M$
λ	Arrival rate of vehicles	$J(z)$	Probability generating function of J
α	Service rate of vehicles	$\mathbb{K}$	Number of tasks served during the lifetime of a VC
R	Residency times of the vehicles in the VC	$\overline{\mathbb{K}}$	Average number of tasks served during the lifetime of a VC
$f_R(r)$	Probability density function of the residency times	$\mathbb{K}(z)$	Probability generating function of $\mathbb{K}$
$R(s)$	Laplace transform of $f_R(r)$	γ	Average number of tasks in a job
T	Interarrival times of the vehicles at the VC	L	Number of tasks in a job
$f_{\mathrm{T}}(\tau)$	Probability density function of the interarrival times	L_i	Number of tasks in the i^{th} job
M	Number of customers served during the busy period of a $G/G/\infty$ queue	$L(z)$	Probability generating function of L
$\overline{M}$	Average number of customers served during the busy period of a $G/G/\infty$ queue	P_0	Probability that there is no task in a job
$M(z)$	Probability generating function of M	A_n	Number of tasks in n jobs
$\widehat{M}$	Number of vehicles served during the busy period of a $M/G/\infty$ queue	$A_n(z)$	Probability generating function of A_n
$\widehat{M}(z)$	Probability generating function of $\widehat{M}$	$\mathbb{N}$	Number of completed jobs during the lifetime of the VC
J	Number of completed tasks by a single vehicle	$\overline{\mathbb{N}}$	Average number of completed jobs during the lifetime of the VC
μ	Service rate of a task	J_i	Number of tasks served by the i'th vehicle, $i = 1 \dots M$
λ	Arrival rate of vehicles	$J(z)$	Probability generating function of J
α	Service rate of vehicles	$\mathbb{K}$	Number of tasks served during the lifetime of a VC
R	Residency times of the vehicles in the VC	$\overline{\mathbb{K}}$	Average number of tasks served during the lifetime of a VC
$f_R(r)$	Probability density function of the residency times	$\mathbb{K}(z)$	Probability generating function of $\mathbb{K}$

Table 4.1. Main notations of Chapter 4

4.1.3. Mathematical assumptions

We will be considering free-flow highway scenario where vehicles' driving behaviors are independent of each other. It will be assumed that the residency times of the vehicles in the VC are i.i.d random variables with mean $1/\alpha$ and pdf $f_R(r)$, whose Laplace transform is $R(s)$. We assume that the interarrival time of the vehicles to the VC has the pdf $f_{\mathrm{T}}(\tau)$ and the arrival rate of the vehicles is λ.

The main notations employed in this chapter is shown in Table 4.1.

4.2. Derivation of the Computing Capacity

In this section, we will determine the number of completed jobs during the lifetime of a VC. We first determine the number of vehicle arrivals to the VC during its lifetime and the number of tasks each vehicle can serve during its residency. Then, we determine the total number of completed tasks and, subsequently, the number of completed jobs during the lifetime of the VC.

4.2.1. Probability generating function of the number of vehicle arrivals to the VC during its lifetime

We will assume that the interarrival time of the vehicles to VC and residency time the of the vehicles in the VC are i.i.d and they have arbitrary distributions. We will model the number of vehicles in the VC as a $G/G/\infty$ queueing system. The lifetime of the VC corresponds to the busy period of the queueing system. As a result, the number of vehicles that has joined to the VC during its lifetime corresponds to the number of customers served during the busy period of a $G/G/\infty$ queueing system. Let M denote the number of customers served during the busy period of a $G/G/\infty$ queueing system and $M(z)$ its PGF. Then, from Theorem 4 in [123], $M(z)$ is given by,

$$M(z) = \sum_{n=1}^{\infty} z^n \int_0^{\infty} \dots \int_0^{\infty} \prod_{i=1}^{n-1} \left\{ \left[f_R\left(\sum_{j=i}^{n} x_j \right) - f_R(x_i) \right] df_{\mathrm{T}}(x_i) \right\} f_R(x_n) df_{\mathrm{T}}(x_n) \tag{4.1}$$

Let $\widehat{M}(z)$ denote the PGF of the number of customers served during the busy period of a $M/G/\infty$ queue. Then from Theorem 2 in [124], the above $M(z)$ reduces to,

$$\widehat{M}(z) = 1 - \frac{1}{\lambda F(z)} \tag{4.2}$$

where

$$F(z) = \int_{t=0}^{\infty} e^{\left[-\lambda t + \lambda \int_{y=0}^{t} zR(y)dy\right]} dt \tag{4.3}$$

4.2.2. PMF of the number of completed tasks during the lifetime of a VC

Next, we will derive the number of tasks served during the lifetime of the VC. First, we will determine the number of tasks a vehicle serves during its residency time in the VC. Since the execution times of the tasks are i.i.d exponentially distributed with parameter μ, the number of tasks that a vehicle can serve is given by the number of Poisson points with parameter μ that may occur during its residency time. Then, the distribution of the number of completed tasks by a single vehicle, J, is given by

$$P(J = j) = \int_{0}^{\infty} \frac{(\mu r)^j}{j!} e^{-\mu r} f_R(r) dr \tag{4.4}$$

Let us denote the probability-generating function of J as $J(z)$, then

$$J(z) = \sum_{j=0}^{\infty} P(J = j) z^j = \sum_{j=0}^{\infty} \int_{0}^{\infty} \frac{(\mu r)^j}{j!} e^{-\mu r} f_R(r)\, dr\, z^j \tag{4.5}$$

Interchanging the order of summation and integration gives

$$J(z) = \int_{0}^{\infty} e^{-(\mu - \mu z)r} f_R(r)\, dr = R(s)|_{s=\mu-\mu z} \tag{4.6}$$

where $R(s)$ is the Laplace transform of the residency time.

Let $\mathbb{K}$ denote the number of tasks served during the lifetime of a VC, and $\mathbb{K}(z) = E\left[z^{\mathbb{k}}\right]$ denote the PGF of the distribution of $\mathbb{K}$. Let also J_i denote the number of tasks served by the i'th vehicle, $i = 1 \ldots M$, where M is the number of vehicles that were members of the VC during its lifetime. Then, $\mathbb{K}$ is given by,

$$\mathbb{K} = \sum_{i=1}^{M} J_i \tag{4.7}$$

As M and J_i are independent, the PGF of $\mathbb{K}$ is given by,

$$\mathbb{K}(z) = M(z)|_{z=J(z)} \tag{4.8}$$

4.2.3. Probability distribution of the number of completed jobs during the lifetime of VC

Next, we will determine the number of jobs the VC will serve during its lifetime. We assume that the number of tasks in a job is given by the Poisson distribution with parameter γ and each job has at least one task. Let L denote the number of tasks in a job and $L(z)$ the PGF of the distribution of L. Then, we have

$$Prob(L = i) = \begin{cases} \dfrac{1}{1 - P_0}\dfrac{\gamma^i P_0}{i!}, & i \geq 1 \\ 0, & i = 0 \end{cases} \tag{4.9}$$

$$L(z) = E[z^L] = \frac{e^{-\gamma(1-z)} - P_0}{1 - P_0} \tag{4.10}$$

where $P_0 = e^{-\gamma}$.

Let A_n denote the number of tasks in n jobs and $A_n(z)$ the PGF of A_n. Since the number of tasks in each job is i.i.d, from (4.10), we can write the PGF of the sum of the tasks in n jobs, $A_n(z)$, as

$$A_n(z) = [L(z)]^n = \frac{1}{(1-P_0)^n}[e^{-\gamma(1-z)} - P_0]^n$$
$$= \frac{1}{(1-P_0)^n}\sum_{k=0}^{n}\binom{n}{k} e^{-\gamma(1-z)k}(-P_0)^{(n-k)} \tag{4.11}$$

where the last equation in the above follows from the Binomial theorem. We also note that

$$e^{-\gamma(1-z)k} = \sum_{\ell=0}^{\infty}\frac{e^{-\gamma k}(\gamma k)^{\ell}}{\ell!}z^{\ell} \tag{4.12}$$

Substitute (4.12) into (4.11), then we have,

$$A_n(z) = \sum_{\ell=0}^{\infty}\left[\frac{1}{(1-P_0)^n}\sum_{k=0}^{n}\binom{n}{k}\frac{e^{-\gamma k}(\gamma k)^{\ell}}{\ell!}(-P_0)^{(n-k)}\right]z^{\ell} \tag{4.13}$$

From the definition of the probability-generating function, the probability distribution of A_n can be determined from (4.13) as:

$$Prob(A_n = \ell) = \begin{cases}\frac{1}{(1-P_0)^n}\sum_{k=0}^{n}\binom{n}{k}\frac{e^{-\gamma k}(\gamma k)^{\ell}}{\ell!}(-P_0)^{(n-k)}, & \ell \geq n \\ 0, & \ell < n\end{cases} \tag{4.14}$$

Finally, let $\mathbb{N}$ denote the number of completed jobs during the lifetime of the VC and L_i the number of tasks in the i^{th} job. Since the number of tasks in jobs is i.i.d, then the probability distribution of L_i is given by (4.10). Then, the probability that there will be n completed jobs given $\mathbb{k}$ completed tasks when $\mathbb{k} \geq n$ will be given by

$$
\begin{aligned}
Prob(N = n|\mathbb{K} = \Bbbk) &= Prob(A_n \le \Bbbk \cap A_{n+1} > \Bbbk) \\
&= \sum_{i=n}^{\Bbbk} Pr(A_n = i) Pr(L_{n+1} > \Bbbk - i) \\
&= \sum_{i=n}^{\Bbbk} Pr(A_n = i) \sum_{j=\Bbbk-i+1}^{\infty} Pr(L_{n+1} = j) \\
&= \sum_{i=n}^{\Bbbk} Pr(A_n = i) \left(1 - \sum_{j=1}^{\Bbbk-i} Pr(L_{n+1} = j) \right)
\end{aligned} \tag{4.15}
$$

We note that $Pr(\mathbb{N} = n|\mathbb{K} = \Bbbk) = 0$ for $\Bbbk < n$. From the inversion of $\mathbb{K}(z)$ in (4.8), we can determine the probability distribution of $\mathbb{K}$, with which, we can determine the unconditional distribution of the number of completed jobs during the lifetime of a VC, $\mathbb{N}$,

$$
Pr(\mathbb{N} = n) = \sum_{\Bbbk=n}^{\infty} Pr(\mathbb{N} = n|\mathbb{K} = \Bbbk) Pr(\mathbb{K} = \Bbbk) \tag{4.16}
$$

4.2.4. Average number of attempted jobs during VC lifetime

Although we can determine the average number of completed jobs from (4.16), we will provide an alternative derivation of the average number of completed jobs which is computationally less intensive. The work in [125] provides a closed-form result for the average number of customers served during the busy period of an $G/G/\infty$ queue. Thus from equation (2.6) in [125], the average number of vehicles that became a member of a VC during its lifetime, $\bar{M}$, is approximately given by,

$$
\bar{M} \cong \frac{e^{\left(\varphi(C_s^2+1)\right)}(\varphi(C_s^2+1)+1) + \varphi(C_s^2+1) - 1}{2\varphi(C_s^2+1)} \tag{4.17}
$$

where φ and C_s are the mean number of vehicles in the system and the coefficient of variance of the residency time, respectively. We note

$$
\varphi = \frac{\lambda}{\alpha}, \qquad C_s = \frac{\sqrt{E[R^2] - (E[R])^2}}{E[R]} \tag{4.18}
$$

Then, the average number of completed tasks, $\bar{\mathbb{K}}$, is given by,

$$\bar{\mathbb{K}} = \bar{M}E[J] = \bar{M}J'(z)\,|_{z=1} \tag{4.19}$$

Where $E[J]$ is the average number of tasks served to completion by a vehicle. $J'(z)$ is the derivative of $J(z)$ wrt z, which is the PGF of the number of completed tasks by a vehicle during its residency given by (4.5). Let us next determine the average number of tasks in a job. From (4.9),

$$\bar{L} = \frac{\gamma}{1 - P_0} \tag{4.20}$$

Finally, the average number of completed jobs during the lifetime of a VC is given by

$$\bar{\mathbb{N}} = \frac{\bar{\mathbb{K}}}{\bar{L}} \tag{4.21}$$

4.3. Numerical results

In this section, we will present numerical results from the analysis above as well as simulation results to demonstrate that the analysis is correct. We assume that the interarrival and residency times of the vehicles are exponentially distributed. In the base case, the values of system parameters are set as α = 0.2 vehicle/min, μ = 1/3 tasks/ min, λ = 1 vehicle/min, γ = 4 tasks.

Fig. 4.2 to Fig. 4.4 plot the average number of job completions as one of the parameters in the base case varies. In general, the system performance confirms our intuition of the system. An increasing λ leads to more job completions as there will be more vehicles during the lifetime of a VC to execute tasks, as shown in Fig. 4.2. An increasing α or a shorter average residency time reduces the number of tasks that a vehicle completes. Thus, the number of completed jobs will be less, which is shown in Fig. 4.3. An increase in the average number of tasks in a job, γ, will decrease the number of completed jobs, as shown in Fig. 4.4. Finally, it may be seen that numerical and simulations results agree with each other.

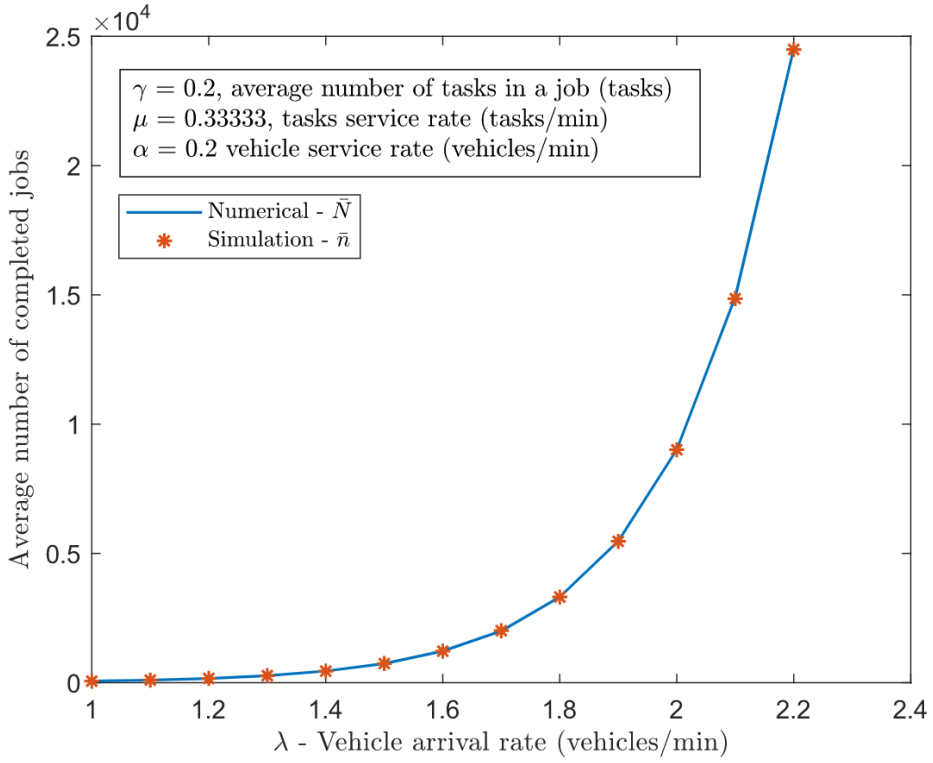


Fig. 4.2. The average number of jobs completed as a function of vehicle arrival rate

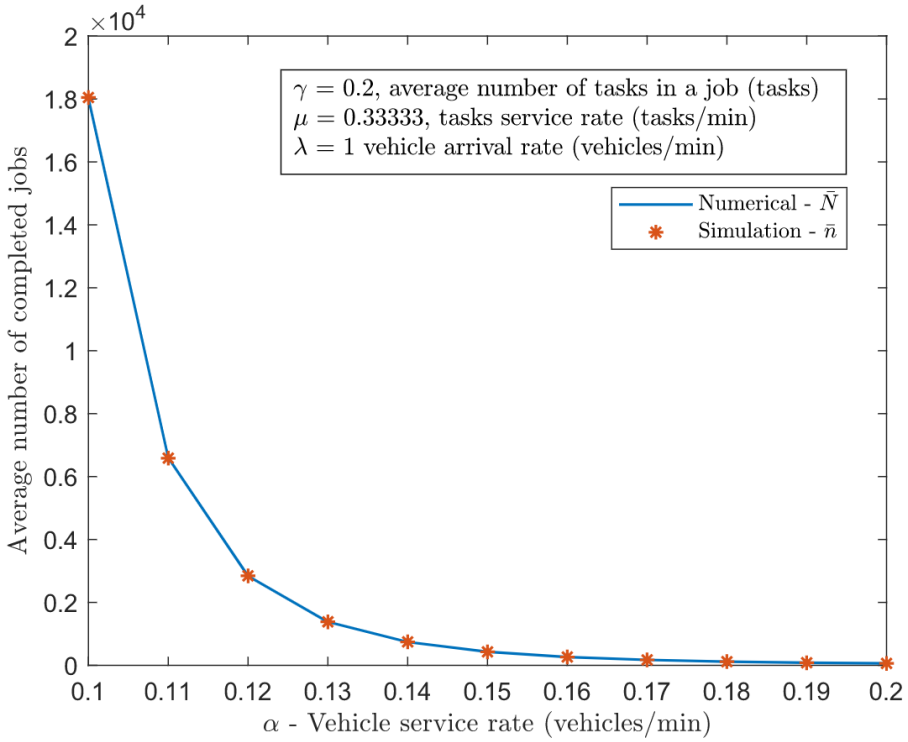


Fig. 4.3. Average number of completed jobs as a function of vehicle service rate.

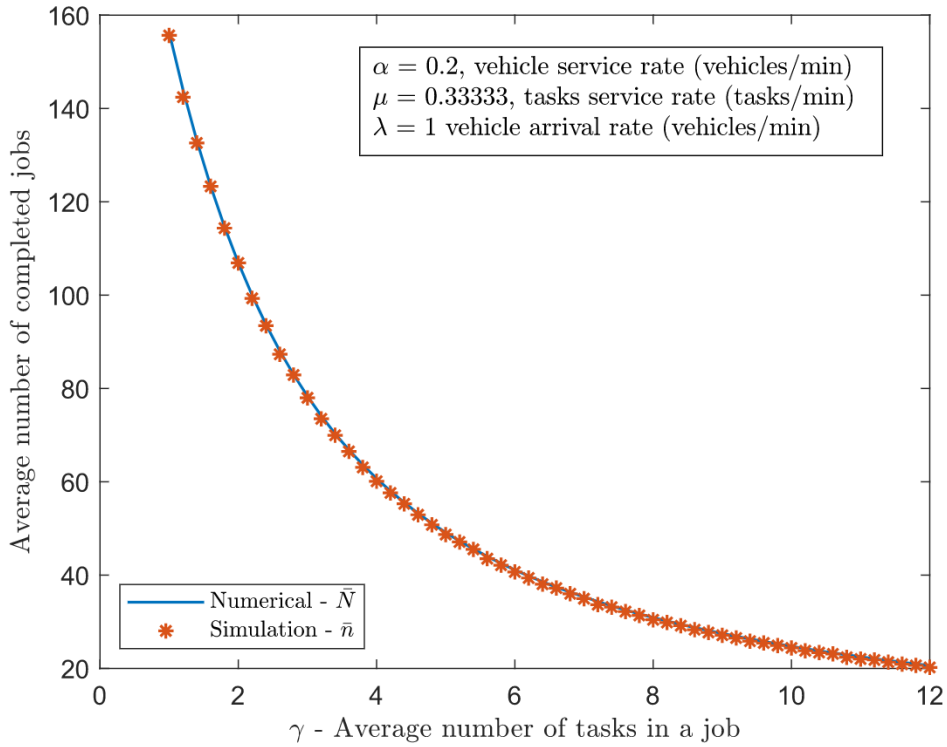


Fig. 4.4. Average number of jobs completed as a function of average number of tasks in a job.

4.4. Summary

In this chapter, we analyze the computing capacity of a VC architecture. Similar to the previous chapters, the VC is formed by vehicles on a free-flow highway. However, in this chapter, when the leader leaves the VC, the leadership is transferred to another member without disconnecting the remaining members. Furthermore, a vehicle can join the VC if it allows the VC to utilize its computing resources and is within the communication range of the leader of the VC. The VC exists when at least one vehicle is in the VC and terminates when the last vehicle disconnects from the VC. The number of completed jobs characterizes the computing capacity of the VC during its existence. We model the number of vehicles within the VC as a $M/G/\infty$ queueing system. The number of tasks in a job follows a Poisson distribution and task execution times are exponentially distributed. Under these assumptions, we first determine the probability distribution of the total number of vehicles that join the VC during its existence. Then, we follow up with the distribution of the number of tasks served by these vehicles. Finally, we derive the distribution of the number of jobs completed during the VC lifetime.

Chapter 5

Computing capacity of a robotaxi fleet

Future vehicles will likely become increasingly autonomous for various reasons, such as comfort, safety, and economics. In fact, besides traditional modes of transportation, passengers will be able to order autonomous taxis or robotaxis operated by technology companies to get to their destinations. From the perspective of the robotaxi fleet operators, they would want to utilize all the resources of their fleet fully and continuously. However, passengers' demand for robotaxi services will be high during rush hours and low during off-peak periods. Thus, the operators are left with an intermittently idling fleet of computationally powerful autonomous vehicles. This work explores the possibility of utilizing the processing units on these vehicles as a computing cluster during their idling period and providing computing services to nearby external customers with computationally demanding tasks.

Computing cluster based on robotaxis is a form of edge computing (EC) and, therefore, holds several advantages over cloud computing. For example, a robotaxi fleet can execute tasks with lower latency than cloud computing due to its proximity to end users. Additionally, as developers must design robotaxis to be fault-tolerant to ensure passengers' safety, the onboard computing units should also be relatively reliable. This characteristic of robotaxi-based EC will ensure the ease of deployment and operation of such computing clusters. Therefore, the feasibility of such a system should be higher than its counterparts.

We study the performance of the system under two scenarios, infinite and finite backlog of tasks. In the infinite backlog case, there will always be tasks to execute for idling taxis. In this case, we derive the probability distribution of the number of tasks that the fleet can serve to completion during a cycle. A cycle is defined as the interval between two consecutive time points when the entire fleet becomes idle. In the finite backlog case, we assume that the tasks requiring service arrive at the system according to a Poisson process. The tasks requiring service enter a queue until they can be served by an idling taxi. If a task is pre-empted during execution

due to the arrival of a passenger service request, it has to wait until another idling taxi becomes available. We derive an approximation for the average task delay in the system.

5.1. System model and assumptions

5.1.2. Robotaxi model

We consider a robotaxi company that owns a fleet of vehicles. The fleet operates in a geo-fenced area for regulatory and efficiency reasons and serves only passenger requests within that area. To manage and assist the operations of the vehicles of the fleet, the fleet operator connects wirelessly to the vehicles. The operator monitors the occupancy status of each vehicle and assigns the passenger requests and computing tasks over the air to these vehicles. In addition, the fleet operator also manages task transfer when the arrival of a passenger request halts the execution of a task. Thus, the operator will know when the vehicles are transporting passengers or are idling and ready to execute tasks. The vehicle's only responsibility is carrying out assigned passenger requests, executing tasks, and backing up the execution progress when interruptions occur. From here on, the vehicles in the robotaxi fleet are called taxis.

We will assume that the arrival of the passenger requests is according to a Poisson process with parameter λ. The passenger service requests will wait in a queue until taxis become available to serve them. The passenger service times are assumed to be exponentially distributed with parameter α. The fleet will include c taxis in total. As a result, we will model the service of the passengers as a $M/M/c$ queuing system.

5.1.3. Task service model

The execution times of the tasks are independent of passenger service times, and they are assumed to be exponentially distributed with the parameter μ. When a taxi in the fleet finishes serving a passenger, if there are no pending passenger requests for service, it will idle and will be available for execution of computing tasks. During this period, the taxi will execute tasks assigned by the fleet operator. However, since the main functionality of the robotaxi fleet is to provide transportation services, carrying out passenger requests from taxi hailers are prioritized over executing tasks. Thus, if a passenger service request arrives when an idling taxi is running a

task, the taxi will halt the execution, save the progress, and drive to the passenger location to serve it. The operator will retrieve the unfinished task with the progress intact and assign it to another idling taxi. When the taxi completes the trip, it will continue to serve other waiting requests from the passengers, if any. Otherwise, if there are no waiting passenger requests, it will be available for task execution.

We will consider two operating scenarios. In the first scenario, the system has a seemingly infinite backlog of tasks to execute; in the second scenario, the backlog of tasks will be finite. In the second scenario, we will assume that arrival of task requests for service is according to a Poisson process with rate β. In the following, we will evaluate the task-serving performance of the system for both scenarios. We will derive the distribution of the number of tasks the system can serve over a period for the first scenario and the average task delay in the system for the second scenario.

In Fig. 5.1a, we show the above task service strategy for a robotaxi fleet with 2 taxis for scenario 1. When passenger 1 arrives at the system, taxi 1 begins to serve this passenger. As there are no other pending passenger requests for service at this time, taxi 2 begins simultaneously executing task 1. However, during the task execution, the passenger 2 request arrives. As a result, taxi 2 has to halt its task execution and start serving passenger 2. Taxi 2 returns the task to the operator with all its progress intact. When taxi 1 finishes serving passenger 1, since there are no pending passenger requests it resumes execution of task 1 from where it was halted. The same thing happens to task 2 where its execution is halted by the arrivals of passengers 3 and 5.

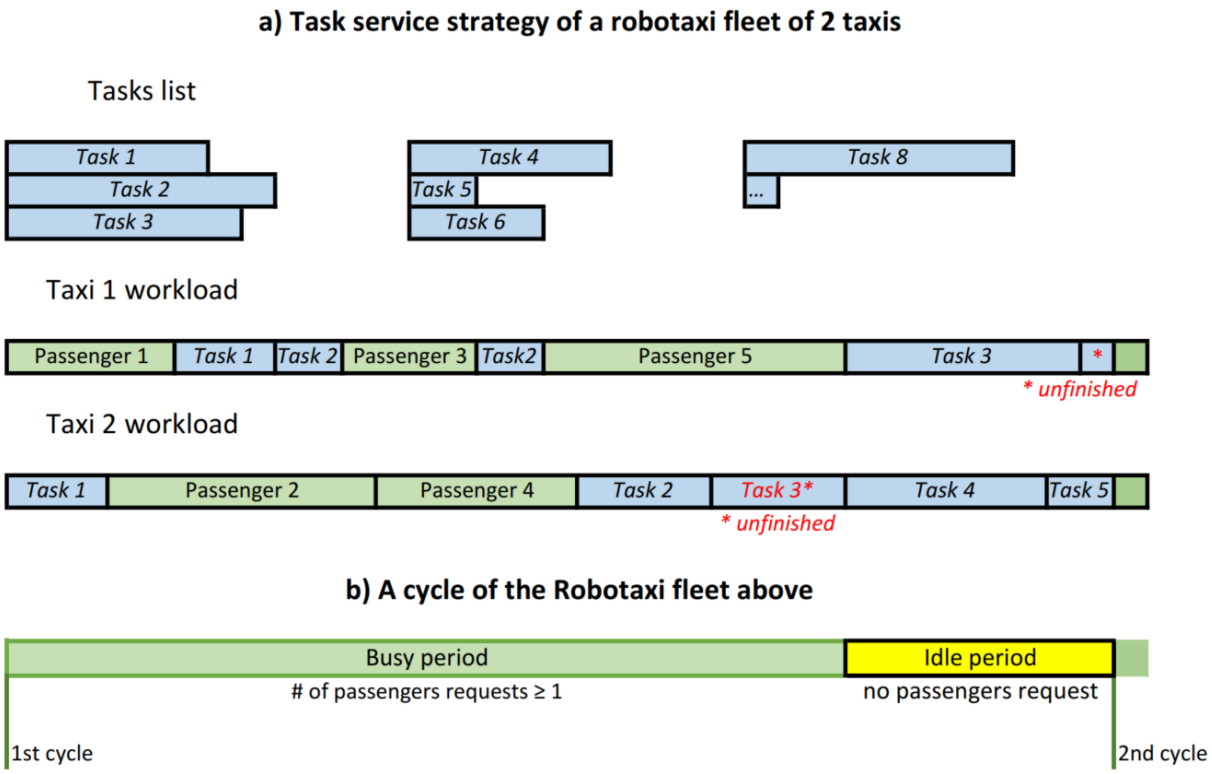


Fig. 5.1. An example of the passenger and task service strategy mechanism of a robotaxi fleet with 2 taxis for scenario 1. Additionally, the figure also shows the relationship between the number of passengers in the system and the duration of the busy period, the idle period and the cycle.

5.1.4. Cyclical nature of the system

We will refer to the period that all the taxis are idle, meaning none of them serving any passenger requests, as system idle period. The system idle periods will alternate with system busy periods. During the system busy period at least one taxi will be busy serving a passenger request at any time. A system idle period followed by a system busy period will be referred to as a cycle. During the system busy period some taxis will be idle some of the time. We will reserve “idle period” to refer to the total amount of time that a single taxi will be idle during a system busy period. We will consider operation of a taxi fleet in cycles. During a system idle period all the taxis will be executing tasks. During the system busy period, taxis will be executing tasks during their idle periods. During the system busy period the execution of the tasks will be

intermittent and often interrupted to serve passenger requests. An example of the cycle is shown in Fig. 5.1b.

The main notations used in this chapter is shown in Table 5.1.

Notation	Description	Notation	Description
c	Number of taxis in the robotaxi fleet	X	Length of the idle period
μ	Service rate of a task	Q_b	Number of tasks the system can compute during the busy period
λ	Arrival rate of passengers	Q_i	Number of tasks the system can compute during the idle period
α	Service rate of a passenger	$\mathbb{K}$	Tasks served during the idle period
L	Interval of the busy period	Q	Total number of tasks the system finish during the busy cycle
$\varphi_i(s)$	Laplace transform of the first passage time from state i to state 0 in a $M/M/c$ queue	k	Number of passengers in the system
$\Psi(s)$	Laplace transform of the length of the busy period of an $M/M/1$ queue with service rate $c\alpha$ and arrival rate λ.	j	Number of idle taxis in the system
N_b	Number of passengers served during the busy period	m	Number of idle taxis serving tasks in the system
$\phi_i(z)$	PGF of the number of passengers served during the first passage time from state i to state 0	n	Number of tasks in the system
$\Phi(z)$	PGF of the number of passengers served during the busy period of an $M/M/1$ queue with service rate $c\alpha$ and arrival rate λ.	$\bar{d}$	Average delay of tasks in the system
ω	Service time of a passenger in the fleet	X	Length of the idle period
Y_b	Length of the busy period	Q_b	Number of tasks the system can compute during the busy period
Y_p	Total time that taxis spend serving passengers during the busy period.	Q_i	Number of tasks the system can compute during the idle period
Y_u	Idle time of the taxis during the busy period	$\mathbb{K}$	Tasks served during the idle period
c	Number of taxis in the robotaxi fleet	Q	Total number of tasks the system finish during the busy cycle
μ	Service rate of a task	k	Number of passengers in the system
λ	Arrival rate of passengers	j	Number of idle taxis in the system
α	Service rate of a passenger	m	Number of idle taxis serving tasks in the system
L	Interval of the busy period	n	Number of tasks in the system

Table 5.1. Main notations of Chapter 5

5.2. Computing Capacity of a Robotaxi Fleet with Infinite Backlog of Tasks

In this section, we determine the computing capacity of the system described above for an infinite backlog of tasks. Since fleet operation is in cycles, we will derive the probability distribution of the number of tasks that can be served to completion during a cycle and its first moment. For this result, we need to determine the number of tasks that can be served during system idle and busy periods. For the latter, we would need to know the duration of a system busy period and the number of passengers served during that busy period.

5.2.1. Laplace transform of the duration of a busy period of a $M/M/c$ queue

Next, we will determine the system busy period of a robotaxi fleet. As stated in the previous section, the passenger service may be modeled as a $M/M/c$ queuing system. Let φ_i, $\varphi_i(s)$ denote the duration of the first passage time from state i to state 0 and its Laplace transform in a $M/M/c$ queue. As a result, the duration of the busy period of the $M/M/c$ queue and its Laplace transform are given by φ_1 and $\varphi_1(s)$, respectively. In [126], the authors derived a recursion to determine $\varphi_i(s)$ given below,

$$\varphi_1(s) = \frac{\lambda}{\lambda + \alpha + s}\varphi_2(s) + \frac{\alpha}{\lambda + \alpha + s} \tag{5.1}$$

$$\varphi_i(s) = \frac{\lambda}{\lambda + i\alpha + s}\varphi_{i+1}(s) + \frac{i\alpha}{\lambda + i\alpha + s}\varphi_{i-1}(s), \qquad 2 \le i \le c-1 \tag{5.2}$$

$$\varphi_c(s) = \varphi_{c-1}(s)\Psi(s), \qquad \varphi_0(s) = 1 \tag{5.3}$$

$$\Psi(s) = \frac{\lambda + c\alpha + s - ((\lambda + c\alpha + s)^2 - 4c\lambda\alpha)^{\frac{1}{2}}}{2\lambda} \tag{5.4}$$

We note that $\Psi(s)$ is the Laplace transform of the duration of the busy period of an $M/M/1$ queue with service rate $c\alpha$ and arrival rate λ.

5.2.2. PGF of the number of passengers served during a system busy period

Next, we provide the result for the number of passengers served during a system busy period. Let N_b and $N_b\,(z)$ denote the number of passengers served during a system busy period

and its PGF, respectively. Let $\phi_i(z)$ denote the PGF of the number of passengers served during the first passage time from state i to state 0 in a $M/M/c$ queuing system. Thus, $\phi_1(z)$ will give PGF of the number of customers served during the busy period of an $M/M/c$ queue. As a result, the PGF of the number of passengers served during the system busy period of the robotaxi fleet is given by $N_b\,(z) = \phi_1(z)$. From [126], we have the following recursion to determine $\phi_i(z)$

$$\phi_1(z) = N_b\,(z) = E[z^{N_b}] \tag{5.5}$$

$$\phi_1(z) = \frac{\lambda}{\lambda+\alpha}\phi_2(z) + z\frac{\alpha}{\lambda+\alpha} \tag{5.6}$$

$$\phi_i(z) = \frac{\lambda}{\lambda+i\alpha}\phi_{i+1}(z) + z\frac{i\alpha}{\lambda+i\alpha}\phi_{i-1}(z), \qquad 1 \le i \le c-1 \tag{5.7}$$

$$\phi_c(z) = \phi_{c-1}(z)\Phi(z), \qquad \phi_0(z) = 1 \tag{5.8}$$

$$\Phi(z) = \frac{\lambda + c\alpha - ((\lambda+c\alpha)^2 - 4c\lambda\alpha z)^{\frac{1}{2}}}{2\lambda} \tag{5.9}$$

We note that $\Phi(z)$ is the PGF of the number of passengers served during the system busy period of an $M/M/1$ queue with service rate $c\alpha$ and arrival rate λ.

5.2.3. Computing capacity of the robotaxi fleet

Now that we have the duration of the system busy period and the number of passengers served during that period, we will be able to determine probability distribution of the number of tasks that may be served during a cycle of the robotaxi fleet. The number of tasks served during a cycle is given by the sum of the number of tasks served during system busy and idle periods.

First, we will determine the number of tasks served during a system busy period. For this, we will determine sum of the durations of idle periods of all the taxis during a system busy period. Let Y_p denote the total amount of time that taxis spend serving the passengers during a system busy period and Y_p(s) its Laplace transform. Also, let Y_u denote the sum of the durations of idle periods of all the taxis during a system busy period and $Y_u(s)$ its Laplace transform. From these definitions, we have,

$$Y_u = Y_b - Y_p \tag{5.10}$$

Where $Y_b = c\varphi_1$. The Laplace transform of Y_b is given by,

$$Y_b(s) = E[e^{-sY_b}] = E[e^{-sc\varphi_1}] = \varphi_1(s)|_{s=sc} \tag{5.11}$$

Let ω_i denote the service time of a passenger. Then,

$$Y_p = \sum_{i=1}^{N_b} \omega_i \tag{5.12}$$

Since passenger service times are i.i.d, we let ω denote ω_i. Then, from (5.5) we have,

$$Y_p(s) = N_b(z)]_{z=\omega(s)} \tag{5.13}$$

Where, $\omega(s)$ is the Laplace transform of a passenger's service time given by,

$$\omega(s) = \frac{\alpha}{s+\alpha} \tag{5.14}$$

Let $f_{Y_b}(y_b), f_{Y_p}(y_p)$ and $f_{Y_u}(y_u)$ denote the pdfs of the random variables Y_b, Y_p, Y_u, respectively. We can obtain $f_{Y_b}(y_b)$ and $f_{Y_p}(y_p)$ through the inverse Laplace transform of $Y_b(s)$ and $Y_p(s)$ from (5.11) and (5.5), respectively. Since $Y_b > Y_p$, from (5.10), we can write

$$f_{Y_u}(y_u) = \int_0^\infty f_{Y_b}(y_u + y_p) f_{Y_p}(y_p) dy_p \tag{5.15}$$

By taking the Laplace transform of the above pdf, we obtain $Y_u(s)$. During the idling times of the system busy periods, taxis will be executing tasks. Since task execution times are exponentially distributed with parameter μ, the number of tasks that can be executed to completion will be given by the number of Poisson points that may occur with parameter μ during the Y_u interval. From equation 5.46 in [121], PGF of the number of tasks the system can compute during the system busy period, $Q_b(z)$, is given by,

$$Q_b(z) = Y_u(s)|_{s=\mu-\mu z} \tag{5.16}$$

Next, we will derive the number of tasks that can be executed during the system idle period. During the system's idle period, there are no passengers in the system. Since the

passengers arrive at the system according to a Poisson process with parameter λ, the system idle duration will be exponentially distributed with the same parameter. Let random variable X denote the duration of the idle period and $f_X(x)$ its pdf, then,

$$f_X(x) = \lambda e^{-\lambda x}, \qquad x \geq 0 \tag{5.17}$$

Let us also define $\mathbb{K}$ as the number of tasks served during the system idle period by a single taxi. Also let, $P_{\mathbb{K}}(\Bbbk)$ and $\mathbb{K}(z)$ denote probability distribution and its PGF of $\mathbb{K}$ respectively. If we condition on the duration of the system idle period, the probability distribution of the number of tasks completed during this period,

$$P_{\mathbb{K}}(\Bbbk|X = x) = \frac{(\mu x)^{\Bbbk} e^{-\mu x}}{\Bbbk!}, \qquad \Bbbk \geq 0 \tag{5.18}$$

$$\mathbb{K}(z|X = x) = e^{-\mu x(1-z)} \tag{5.19}$$

Let also Q_i and $Q_i(z)$ denote the sum of the tasks all the taxis serve during the system idle period and its PGF, respectively. We then can write,

$$Q_i(z|X = x) = [K(z|X = x)]^c = e^{-\mu c x(1-z)} \tag{5.20}$$

$$Q_i(z) = \int_0^\infty e^{-\mu c x(1-z)} f_X(x) dx = \frac{\lambda}{\lambda + \mu c(1-z)} \tag{5.21}$$

Finally, let us denote the total number of tasks completed during a cycle as Q. Then Q is the number of tasks completed during the system busy and idle periods, $Q = Q_i + Q_b$. From (5.16) and (5.21), the number of tasks served during a cycle of a robotaxi system is given by,

$$Q(z) = Q_i(z) Q_b(z) \tag{5.22}$$

5.2.4. Mean number of completed tasks during a cycle

The distribution of the number of completed tasks can be challenging to determine numerically for a large number of taxis. On the other hand, its mean is often easier to obtain. As a result, we will next determine the mean number of tasks served to completion during a cycle.

From (5.10), (5.13) and (5.17), we can write the mean for the sum of the idle times of the taxis during the system busy period Y_u as

$$E[Y_u] = E[Y_b] - E[Y_p] = -Y_b'(s)|_{s=0} + Y_p'(s)\big|_{s=0} \tag{5.23}$$

where $Y_b'(s)$ and $Y_p'(s)$ denote the derivatives of $Y_b(s)$ and $Y_p(s)$ w.r.t s. As a computing task is exponentially distributed with parameter μ, then the mean number of completed tasks during the system busy period is given by,

$$E[Q_b] = \mu E[Y_u] \tag{5.24}$$

From (5.21), the mean number of tasks served during a system idle period will be given by,

$$E[Q_i] = \frac{dQ_i(z)}{dz}\bigg|_{z=1} \tag{5.25}$$

Finally, the mean number of tasks served during a cycle of a robotaxi system is given by,

$$E[Q] = E[Q_i] + E[Q_b] \tag{5.26}$$

5.3. Delay analysis of the tasks in a robotaxi fleet with Finite Backlog of Tasks

In this section, we consider the same system model as the previous section, except that we assume a finite backlog of tasks waiting to be served. We assume that the tasks arrive at the system according to a Poisson process with parameter β. As in the previous section, passengers have preemptive priority over tasks. We will determine the average delay of a task in the system. The system may be modeled as a $M/M/c$ queue with two classes of customers. The high-priority customers (passengers) do not see the low-priority customers (tasks). As in the previous section, the service of the passengers follows the $M/M/c$ queuing system. Then from [121], the stationary probability distribution of having k passengers in the system, P_k, is given by,

$$P_k = \begin{cases} P_0 \dfrac{(c\rho)^k}{k!}, & for\ 0 < k < c \\ P_0 \dfrac{(c\rho)^k c^{c-k}}{c!}, & for\ c \le k \end{cases} \tag{5.27}$$

where ρ is the utilization factor given by, $\rho = \frac{\lambda}{c\alpha}$.

In [127], an algorithmic method has been developed to determine the distribution of the number of low-priority customers in the system. However, the application of this method is computationally very cumbersome for more than two servers, $c > 2$. As a result, we will develop an approximate analysis for the low-priority customers. If there are k passengers in the system, then, there will be $j = c - k$ idle taxis for $k \leq c$. From (5.27), we can determine the probability distribution of j idle taxis in the system, T_j, as follows

$$T_j = \begin{cases} P_{c-j}, & for \quad 1 < j \leq c \\ 1 - \sum_{k=0}^{c-1} P_k, & for \ j = 0 \end{cases} \tag{5.28}$$

Furthermore, let us define the average number of idle taxis as m,

$$m = \left\lfloor \sum_{j=0}^{c} jT_j \right\rfloor \tag{5.29}$$

Then, we make the approximation that tasks are served by m taxis continuously. Thus, the service given to the tasks will be modeled as a $M/M/m$ queueing system. Let us further denote the utilization factor of this system as $\sigma = \frac{\beta}{m\mu}$. Then, the stationary distribution of n tasks in the system, Q_n, can be written as

$$Q_n = \begin{cases} Q_0 \frac{(n\sigma)^j}{j!} & for\ 0 < j < n \\ Q_0 \frac{(n\sigma)^j q^{n-j}}{j!} & for\ n \leq j \end{cases} \tag{5.30}$$

From the definition of the expected value, we can determine the average number of tasks in the system $\bar{n}$ from (5.22). Using Little's result, the average delay of the tasks in the system $\bar{d}$ is given by,

$$\bar{d} = \frac{\bar{n}}{\beta} = \frac{\sum_{n=0}^{\infty} nQ_n}{\beta} \tag{5.31}$$

5.4. Numerical results

This section presents numerical results about the analysis for the infinite and finite backlog of tasks. We also provide results of a Monte-Carlo simulation written in Matlab to verify the analysis. Simulate was run 50,000 cycles. We note that simulation run time increases rapidly with increasing duration of busy periods; as a result, we could not obtain simulation results for large values of busy period duration. In each cycle, the passengers arrive at the system according to a Poisson process and either immediately receive service if there are idle taxis or enter a waiting queue until taxis become available. The tasks have exponentially distributed execution times and are served according to the service strategy described in the system model. An arriving passenger may preempt an executing task. The total number of tasks served to completion is averaged over the number of cycles to determine the average number of tasks completed.

Fig. 5.2 and Fig. 5.3 show the average number of completed tasks during a cycle for an infinite backlog of tasks as a function of passenger service and arrival rates, respectively. Fig. 5.3 also shows average number of completed tasks per minute as a function of passenger arrival rate. In Fig. 5.2, the average number of completed tasks decreases as passenger service rate increases because the busy period duration decreases as service rate increases. In Fig. 5.3, we can see that the average number of completed tasks initially slightly decreases until passenger arrival rate equals to $\lambda \approx 2$ vehicles/min and after that, it starts increasing. However, this plot is misleading because the average cycle duration is not constant as passenger rate increases. In the same figure, it's shown that the average number of completed tasks per minute decreases linearly as a function of passenger arrival rate. This behavior may be explained by Fig. 5.4, which shows the average cycle, system busy period, and the total idle period durations of the fleet as a function of passenger arrival rate. It may be seen that while the duration of the busy period increases as passenger arrival rate increases, the system's idle period decreases. As a result, cycle duration initially decreases but then it starts increasing with increasing passenger service rate because the busy period increases much faster than the system idle period decreases. Fig. 5.4 also shows that duration of the fleet's total idle period follows the pattern of cycle duration. The number of completed tasks during a cycle is a function of the total idle period, which explains the behavior of the curve in Fig. 5.3. Though the number of completed tasks increases during a cycle as

passenger arrival rate increases, when normalized to cycle duration, the average number of completed tasks per unit time decreases.

Fig. 5.5 shows the average delay of a task in the case of a finite task backlog. The average task delay has been plotted as a function of task service rate for constant values of passenger and task arrival rates and passenger service rates. The arriving tasks join a queue when there are no idling taxis and can be preempted by any new arriving passengers during their execution. We can see that the average task delay decreases as the service rate increases.

Finally, in all figures, numerical results match simulation results.

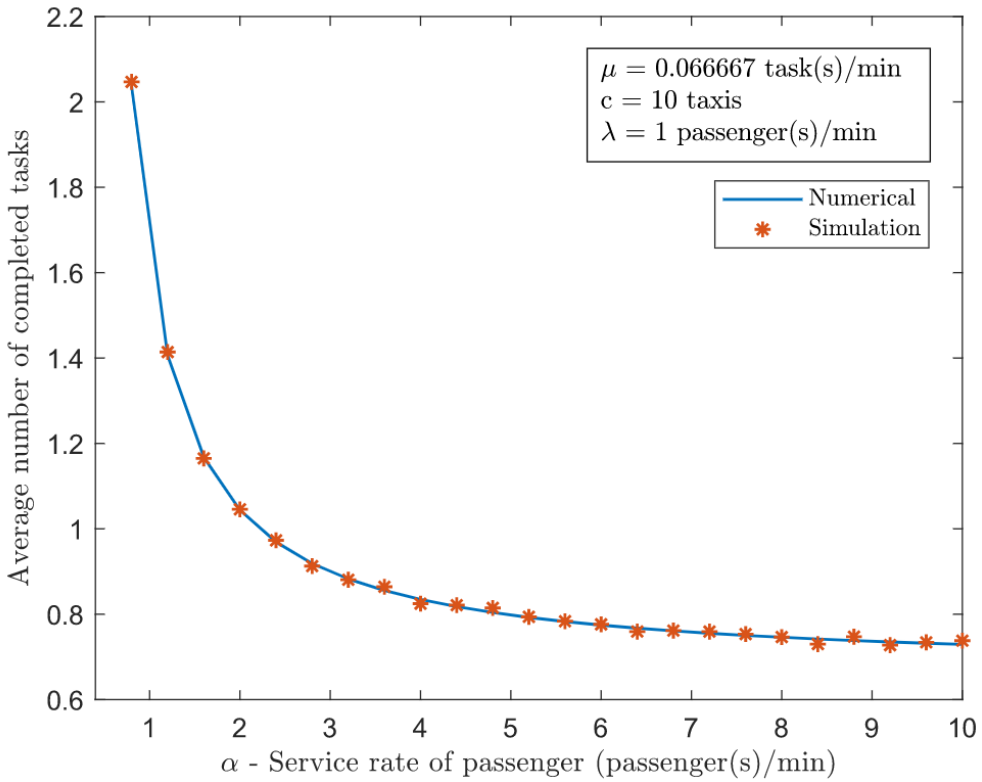


Fig. 5.2. Numerical and simulation results of the average number of completed tasks during a cycle as a function of passenger service rate α

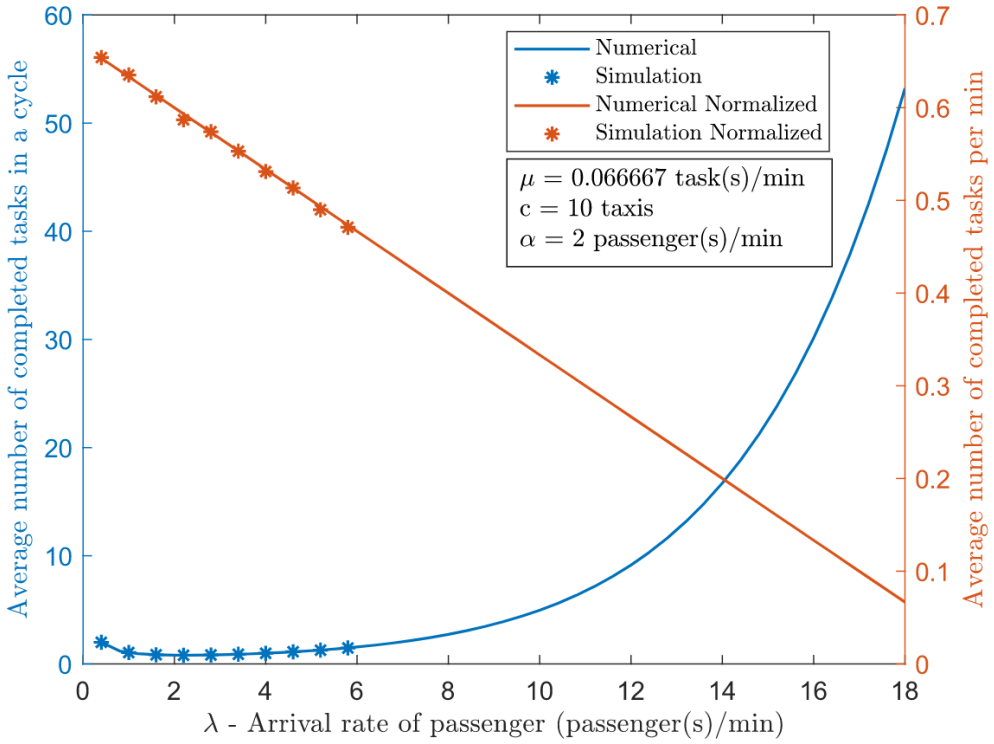


Fig. 5.3. Numerical and simulation results of the average number of completed tasks during a cycle as a function of passenger arrival rate λ

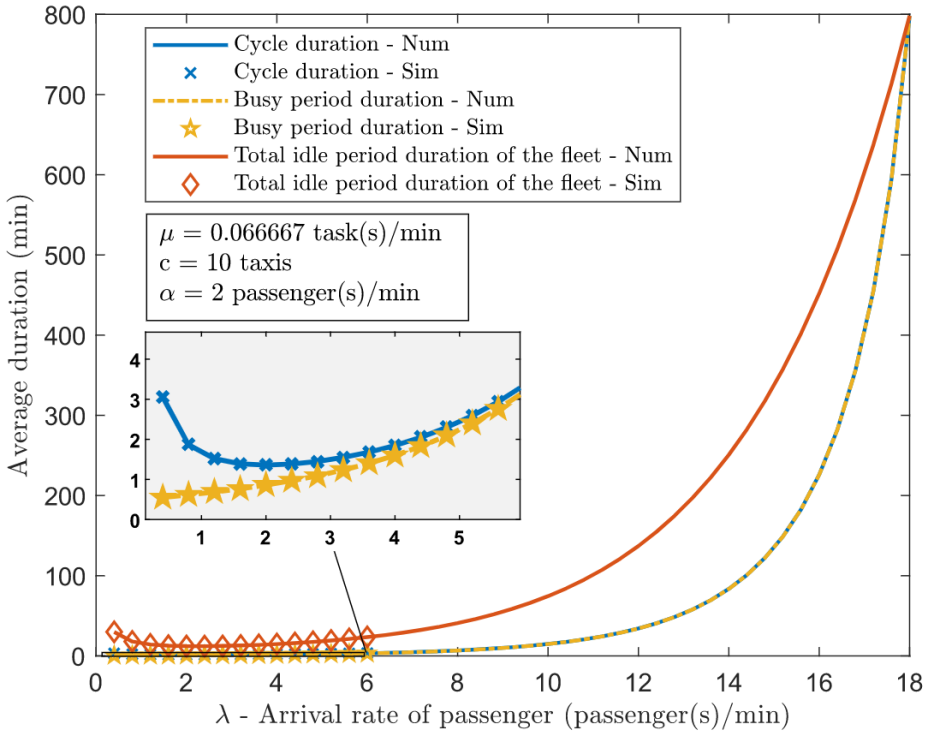


Fig. 5.4. Numerical and simulation results of the average cycle, system busy period and total idle durations of the fleet as a function of the passenger arrival rate.

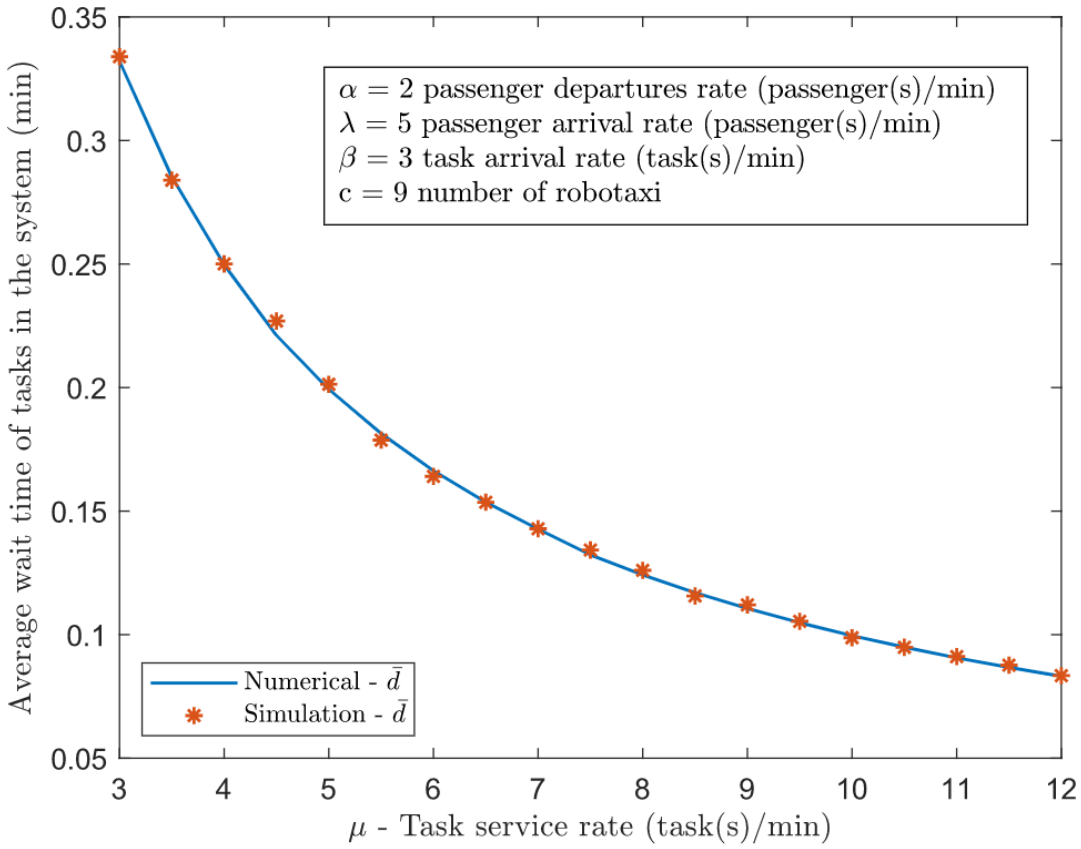


Fig. 5.5. Numerical and simulation results of the average task delay as a function of task service rate.

5.5 Summary

In this chapter, we considered a robotaxi fleet that transports passengers to their destinations. Since the robotaxis are autonomous, they will have high computing resources. Fleet operators would like to utilize these taxis during idle times as they are costly. In this work, we evaluate the fleet's performance in serving computational tasks during idle times under infinite and finite backlog of tasks. We model the system as a $M/M/c$ queue with two types of customers, which are passengers and tasks. Passengers are given preemptive priority over the tasks. In the infinite backlog of tasks, we determine the probability distribution and the first moment of the number of completed tasks during a cycle. We show that the average number of completed tasks decreases with increasing task service rate and passenger arrival rates. In the finite backlog of tasks, the average task delay decreases with an increasing task service rate. These results will help in determining benefits of using idle resources in a robotaxi fleet in task execution.

Chapter 6

Conclusion and Future work

Conclusion

The data generated at the edge of the Internet is increasing rapidly due to the proliferation of smart devices. This data cannot be transferred to the cloud because of high cost and latency. Further, the data needs to be processed in real-time to be of value to the users. Modern vehicles are equipped with powerful computers and storage facilities. Furthermore, these vehicles are being designed with redundancy for vehicular autonomy. As a result, the computing resources of these vehicles can be underutilized, which can be pooled together to form a computing cluster called a vehicular cloud. Vehicular clouds may help to solve the processing needs of the massive data being generated at the edge of the Internet. As a result, there is great interest in determining the capabilities of the VCs.

In this thesis, we model and analyze the performance of Vehicular Cloud under different traffic conditions, task assignment strategies, job models, and service priorities. The goal of this work is to analytically derive the key performance measures, such as job completion time and computing capacity of VCs, to estimate the capability of the VCs. In the following, the main contributions of this work are briefly summarized.

The first contribution concern the derivation of the probability distribution of the job completion time of an ad-hoc vehicular cloud on a highway under free-flowing and congested traffic conditions with a service interruption strategy. In this service strategy, when a vehicle serving a task leaves the VC, that task is assigned to an idle vehicle in the VC. It's assumed that the vehicles arrive at the VC according to a Poisson process, and they have independent residency times under free-flowing traffic and correlated residency times under congested traffic. We assumed that a job contains multiple independent tasks with random execution times. Under this service strategy, we derived the probability density function of job completion time for both mentioned traffic conditions with zero migration overhead. This result is significant because no

service strategy can provide a lower job completion time. We also provide an ad hoc method to include the migration overhead in the job completion time.

In the second contribution, we determined an approximation to the job completion time under the service interruption avoidance strategy. This service strategy avoids the migration overhead due to service interruption. In this service strategy, a vehicle will only be assigned a task if it can finish its execution during its residency time. Further, a vehicle will be assigned the longest of the unassigned tasks that it can complete. Under these conditions, we derived the distribution of the completion time of the longest task. Numerical results show that the completion time of the longest task may be used to approximate the job completion time. However, the completion time of a job will be long if the job has tasks with long execution times. Thus, a hybrid service strategy that serves short-duration tasks according to the service avoidance strategy and it serves long-duration tasks according to the service with interruption strategy will be a better choice.

In the third contribution, we derived the probability distribution of the number of completed jobs during the lifetime of a VC. This work makes the same assumptions as the previous contribution, except it assumes that the number of tasks in a job follows a Poisson distribution. Under these assumptions, the distribution of the total number of vehicles that join the VC during its existence is derived and serves as a foundation for subsequent results. Then, the distributions of the number of tasks completed by these vehicles and the number of jobs completed during the VC lifetime are derived.

In the final contribution of this thesis, we analyzed the computing capacity of a robotaxi fleet that serves passengers. The vehicles in the fleet form a VC, and they can execute tasks during their idle times; however, passengers have pre-emptive priority over task executions. Contrary to the previous contributions, the VC is permanent, and the number of vehicles in the VC is constant as the robotaxi often functions within a geofenced area. We study this system both under infinite and finite backlog of tasks. In the case of an infinite backlog of tasks, we derived the probability distribution of the number of tasks the system can serve over a period. In the second case, it is assumed that the tasks will queue for service, and we derived the average delay of the tasks in the system.

The work in this thesis helps to a better understanding of the capabilities of vehicular clouds. It determines job completion time in terms of the arrival rate of vehicles to the VC, vehicle residence times, number of tasks in a job, and task execution times. This helps in determining whether a VC can meet the QoS requirements of a job.

Future work

There remain potential models stemming from the current model that can be explored in future work. These problems are described below,

Optimal Hybrid Service Strategy

As explained in the above, hybrid service strategy may result in better job performance. Thus, tasks with execution times below a threshold will be served with service interruption avoidance strategy and those above the threshold with service interruption strategy. It will be useful to determine the optimal value of the threshold that will result in the shortest job completion time. This optimal threshold value will depend on the arrival rate of the vehicles, as well as distributions of residency and task execution times.

Heterogeneity in the VC

The current model assumes that the vehicles have identical computational powers. In practice, this may not be true. The computational powers of vehicles may be different. Also, the vehicles may not make available all their computational powers for the external jobs and keep part of it for their local computations. Another variable may be introduced to the system model to capture the computational powers of the vehicles. We will attempt to extend the present analysis to cover this case. However, the state-space explosion will be a problem.

Migration overhead

In the preliminary work, we have developed an ad hoc method to determine an upper-bound for average job completion time with constant migration overhead. The migration overhead may be a random variable as the data rates of the communication links are variable. We would like to extend our model to include migration overhead rather than determining it in an ad-hoc manner. It will be assumed that the residence time of a vehicle will consist of three stages of

Erlangian distribution with different parameters. The first stage will correspond to the downloading of the VM from the VC leader, the second stage execution of a task, and the third stage to the uploading of either the VM or the results of the completed task to the VC leader. In this case, the number of vehicles in the system may be modeled as an $M / G / \infty$ queuing system. Clearly, we will need to increase the dimensions of the state vector that describes the system. Again, we will attempt to extend the current analysis to handle this case. We will also investigate the results of absorption time in Markov chains theory to help in solving this problem.

Deadline driven task execution

The service strategies studied in this thesis donot take into consideration the deadlines for completion of the task execution times. Thus the tasks in addition to the duration of their execution times may also have deadlines for service completion. Defining the task waiting time as the difference between service completion deadline and task execution time, the tasks with smaller waiting times should be given priority in service. This strategy ensures that a task with less waiting time is executed earlier, thus improving the likelihood of all tasks meeting their respective deadlines. The challenge lies in the likely increase in the dimensions of the state vector that describes the system, similar to the case when migration overhead is included. Despite the complexity, it is expected that incorporating this nuanced scheduling approach will enhance the performance of the VC systems. The analytical methods developed for dealing with heterogeneity and migration overhead could potentially be adapted to handle this new feature in the model.

Non-homogeneous traffic flow

The existing models assume a homogeneous traffic flow where vehicles are joining and leaving the VC at a constant rate. However, in real-world scenarios, the traffic flow might not be uniform, especially during peak and off-peak hours. There might be instances where a large number of vehicles join the VC, causing a sudden surge in the available computational power. Conversely, during off-peak hours, the number of vehicles, and therefore the computational power, may drastically reduce. Future work should aim to incorporate non-homogeneous traffic flow into the model. The traffic flow variation of vehicles could be captured by a non-

homogeneous Poisson process where the arrival and departure rates are time-dependent. The modeling of such a system will provide more accurate results to the real-world.

Appendix A

Solving differential-difference equations in 2.1.2 with transform methods

Let us define the probability generating function (PGF) of the joint probability distribution of the number of uncompleted tasks and number of worker vehicles in the system at time *t* as

$$\mathfrak{P}(z_1, z_2, t) = \sum_{j=0}^{\mathfrak{I}} \sum_{k=0}^{\infty} \mathfrak{P}_{j,k}(t) z_1^j z_2^k \tag{A.1}$$

We now multiply both sides of each of the equations (2.2) to (2.5) by ${z_1}^j {z_2}^k$ and sum them over j and k, which results in:

$$\sum_{j=0}^{\mathfrak{I}} \frac{d\mathfrak{P}_{j,0}(t)}{dt} {z_1}^j = \sum_{j=0}^{\mathfrak{I}} \alpha \mathfrak{P}_{j,1}(t) {z_1}^j - \sum_{j=0}^{\mathfrak{I}} \lambda \mathfrak{P}_{j,0}(t) {z_1}^j \tag{A.2}$$

$$\begin{aligned} \sum_{k=1}^{\infty} \frac{d\mathfrak{P}_{\mathfrak{I},k}(t)}{dt} {z_1}^{\mathfrak{I}} {z_2}^k = & \sum_{k=1}^{\infty} \lambda \mathfrak{P}_{\mathfrak{I},k-1}(t) {z_1}^{\mathfrak{I}} {z_2}^k + \sum_{k=1}^{\infty} (k+1)\alpha \mathfrak{P}_{\mathfrak{I},k+1}(t) {z_1}^{\mathfrak{I}} {z_2}^k \\ & - \sum_{k=1}^{\infty} [\lambda + k\alpha + min(\mathfrak{I}, k)\mu] \mathfrak{P}_{\mathfrak{I},k}(t) {z_1}^{\mathfrak{I}} {z_2}^k \end{aligned} \tag{A.3}$$

$$\sum_{j=1}^{\mathfrak{I}-1}\sum_{k=1}^{\infty}\frac{d\mathfrak{P}_{j,k}(t)}{dt}{z_1}^j{z_2}^k = \sum_{j=1}^{\mathfrak{I}-1}\sum_{k=1}^{\infty}\lambda\mathfrak{P}_{j,k-1}(t){z_1}^j{z_2}^k + \sum_{j=1}^{\mathfrak{I}-1}\sum_{k=1}^{\infty}(k+1)\alpha\mathfrak{P}_{j,k+1}(t){z_1}^j{z_2}^k$$
$$-\sum_{j=1}^{\mathfrak{I}-1}\sum_{k=1}^{\infty}[\lambda + k\alpha + min(j,k)\mu]\mathfrak{P}_{j,k}(t){z_1}^j{z_2}^k$$
$$+\sum_{j=1}^{\mathfrak{I}-1}\sum_{k=1}^{\infty}min(j+1,k)\mu\mathfrak{P}_{j+1,k}(t){z_1}^j{z_2}^k \quad (A.4)$$

$$\sum_{k=1}^{\infty}\frac{d\mathfrak{P}_{0,k}(t)}{dt}{z_2}^k = \sum_{k=1}^{\infty}\lambda\mathfrak{P}_{0,k-1}(t){z_2}^k + \sum_{k=1}^{\infty}(k+1)\alpha\mathfrak{P}_{0,k+1}(t){z_2}^k$$
$$-\sum_{k=1}^{\infty}[\lambda + k\alpha]\mathfrak{P}_{0,k}(t){z_2}^k + \sum_{k=1}^{\infty}\mu\mathfrak{P}_{1,k}(t){z_2}^k \quad (A.5)$$

Using the change of variables technique and algebraic manipulations, we can rewrite equations (A.3), (A.4), and (A.5), respectively, as:

$$\sum_{k=1}^{\infty}\frac{d\mathfrak{P}_{\mathfrak{J},k}(t)}{dt}{z_1}^{\mathfrak{J}}{z_2}^k$$
$$= \lambda{z_1}^{\mathfrak{J}}\sum_{k=0}^{\infty}\mathfrak{P}_{\mathfrak{J},k}(t){z_2}^{k+1} + \alpha{z_1}^{\mathfrak{J}}\left[\sum_{k=0}^{\infty}k\mathfrak{P}_{\mathfrak{J},k}(t){z_2}^{k-1} - \mathfrak{P}_{\mathfrak{J},1}(t)\right]$$
$$-\lambda{z_1}^{\mathfrak{J}}\sum_{k=1}^{\infty}\mathfrak{P}_{\mathfrak{J},k}(t){z_2}^k - \alpha{z_1}^{\mathfrak{J}}\sum_{k=1}^{\infty}k\mathfrak{P}_{\mathfrak{J},k}(t){z_2}^k$$
$$-\mu{z_1}^{\mathfrak{J}}\sum_{k=1}^{\infty}min(\mathfrak{J},k)\mathfrak{P}_{\mathfrak{J},k}(t){z_2}^k \quad (A.6)$$

$$\sum_{j=1}^{\mathfrak{J}-1}\sum_{k=1}^{\infty}\frac{d\mathfrak{P}_{j,k}(t)}{dt}{z_1}^j{z_2}^k=\lambda\sum_{j=1}^{\mathfrak{J}-1}\sum_{k=0}^{\infty}\mathfrak{P}_{j,k}(t){z_1}^j{z_2}^{k+1}$$
$$+\alpha\left[\sum_{j=1}^{\mathfrak{J}-1}\sum_{k=0}^{\infty}k\mathfrak{P}_{j,k}(t){z_1}^j{z_2}^{k-1}-\sum_{j=1}^{\mathfrak{J}-1}\mathfrak{P}_{j,1}(t){z_1}^j\right]-\lambda\sum_{j=1}^{\mathfrak{J}-1}\sum_{k=1}^{\infty}\mathfrak{P}_{j,k}(t){z_1}^j{z_2}^k$$
$$-\alpha\sum_{j=1}^{\mathfrak{J}-1}\sum_{k=1}^{\infty}k\mathfrak{P}_{j,k}(t){z_1}^j{z_2}^k-\mu\sum_{j=1}^{\mathfrak{J}-1}\sum_{k=1}^{\infty}min(j,k)\mathfrak{P}_{j,k}(t){z_1}^j{z_2}^k$$
$$+\mu\sum_{j=1}^{\mathfrak{J}-1}\sum_{k=1}^{\infty}min(j+1,k)\mathfrak{P}_{j+1,k}(t){z_1}^j{z_2}^k \tag{A.7}$$

$$\sum_{k=1}^{\infty}\frac{d\mathfrak{P}_{0,k}(t)}{dt}{z_2}^k$$
$$=\lambda\sum_{k=0}^{\infty}\mathfrak{P}_{0,k}(t){z_2}^{k+1}+\alpha\left[\sum_{k=0}^{\infty}k\mathfrak{P}_{0,k}(t){z_2}^{k-1}-\mathfrak{P}_{0,1}(t)\right]$$
$$-\lambda\sum_{k=1}^{\infty}\mathfrak{P}_{0,k}(t){z_2}^k-\alpha\sum_{k=1}^{\infty}k\mathfrak{P}_{0,k}(t){z_2}^k+\mu\sum_{k=1}^{\infty}\mathfrak{P}_{1,k}(t){z_2}^k \tag{A.8}$$

On the left-hand sides of equations (A.2) to (A.5), since $\sum_{j=0}^{\mathfrak{J}}\sum_{k=0}^{\infty}\frac{d\mathfrak{P}_{j,k}(t)}{\mathrm{dt}}$ converges uniformly, we can interchange the order of differentiation and summation. Then summing the derivations all together results in

$$\sum_{j=0}^{\mathfrak{J}}\frac{d\mathfrak{P}_{j,0}(t)}{dt}{z_1}^j+\sum_{k=1}^{\infty}\frac{d\mathfrak{P}_{\mathfrak{J},k}(t)}{dt}{z_2}^k+\sum_{j=1}^{\mathfrak{J}-1}\sum_{k=1}^{\infty}\frac{d\mathfrak{P}_{j,k}(t)}{dt}{z_1}^j{z_2}^k+\sum_{k=1}^{\infty}\frac{d\mathfrak{P}_{0,k}(t)}{dt}{z_2}^k$$
$$=\sum_{j=0}^{\mathfrak{J}}\sum_{k=0}^{\infty}\frac{d\mathfrak{P}_{j,k}(t)}{dt}z_1^jz_2^k=\frac{\partial}{\partial t}\sum_{j=0}^{\mathfrak{J}}\sum_{k=0}^{\infty}\mathfrak{P}_{j,k}(t)z_1^jz_2^k=\frac{\partial P(z_1,z_2,t)}{\partial t} \tag{A.9}$$

Summing the right-hand sides of equations (A.2), (A.6) to (A.8) and using the results from equation (A.9), we obtain the following partial differential equation of $\mathfrak{P}(z_1,z_2,t)$,

$$\frac{\partial \mathfrak{P}(z_1, z_2, t)}{\partial t} = \lambda z_2 \mathfrak{P}(z_1, z_2, t) + \alpha \frac{\partial \mathfrak{P}(z_1, z_2, t)}{\partial z_2} - \lambda \mathfrak{P}(z_1, z_2, t) - \alpha z_2 \frac{\partial \mathfrak{P}(z_1, z_2, t)}{\partial z_2}$$
$$-\mu \sum_{j=1}^{\mathfrak{I}} \sum_{k=1}^{\infty} min(j,k) \mathfrak{P}_{j,k}(t) {z_1}^j {z_2}^k$$
$$+ \mu \sum_{j=0}^{\mathfrak{I}-1} \sum_{k=1}^{\infty} min(j+1,k) \mathfrak{P}_{j+1,k}(t) {z_1}^j {z_2}^k \quad \text{(A.10)}$$

We note that the above partial differential equation is unsolvable as not all the terms can be written as a function of $\mathfrak{P}(z_1, z_2, t)$ and its partial derivatives.

Appendix B

Section 2.1 Simulation description

In the simulation script, we use a $n \times 5$ matrix to represent the vehicles, such as one in Table B.1, where the rows are the vehicles in the VC during a simulation run, and the columns are the characteristics of the vehicles, such as, unique vehicle ID , minutes until the vehicle's arrival, residency duration of the vehicle in the VC, status of the vehicle, the ID of the task assigned to the vehicle (v_task). Similarly, we use another $\mathfrak{J} \times 5$ matrix to represent the status of the $\mathfrak{J}$ tasks during a simulation run, such as one in Table B.2. The $\mathfrak{J}$ rows of the matrix represent the $\mathfrak{J}$ tasks, and the columns represents the unique task ID, the remaining task execution times, the completion time (t_comp), the status of the tasks (t_stat) and the ID of the vehicle working the task (t_veh).

Vehicle ID	Interarrival times	Residency times	Status	Assigned task ID
1	0	12.65	in_VC	2
2	11.39	16.45	$arriving$	0
...	...	...	...	...

Table B.1. The vehicle matrix upon the start of the simulation where there is 1 initial vehicle

Task ID	Task execution time	Task completion time	Status	Assigned vehicle ID
1	35.86	0	$assigned$	0
2	16.89	0	$suspended$	1
...	...	...	...	...
$\mathfrak{J}$	44.19	0	$suspended$	0

Table B.2. The task matrix upon the start of the simulation where there no time has elapsed and task 2 is assigned to vehicle 1

To record the completion time of all the tasks for all the runs, we create a $n \times \mathfrak{J}$ matrix, where each row is the completion time results for all the runs, and each column is the completion time of task j^{th}. Contrary to the task and vehicle matrices, the values of the completion time matrix will remain intact after each simulation run.

Runs	Task completion times				
	1	2	3	...	$\mathfrak{J}$
1	5.01	3.49	6.32	...	6.6
2	2.99	3.39	1.53	...	4.61
...	...	...	...	...	...
n	1.75	13.6	1.63	...	2.34

Table B.3. The completion time matrix after n runs for a job of $\mathfrak{J}$ tasks

We then create the task matrix by creating a unique ID for the tasks. Then, we generate an exponentially distributed execution times for each task using exprnd function with the mean execution time of a task, $1/\mu$. At this point, there are no vehicles in the VC, thus the remaining columns of the matrix are zero.

Then, we generate the initial vehicles with unique vehicles ID in the VC in MATLAB using poissrnd function with the parameter λ/α. These vehicles will form the first rows of the vehicle matrix. For each of the vehicles, we assign them an exponentially distributed random variable from exprnd function using the mean residency time, $1/\alpha$. As these are initial vehicles, their arrival times to the VC are 0. For each vehicle, we will randomly select the ID of an unassigned task in the task matrix and assign the value to the v_task field. Concurrently, we also change the task status after each assignment. Finally, an arriving vehicle is added to the matrix by generating the interarrival time and residency time to the last row. When the number of initial vehicles is zero, the arriving vehicle is the first row of the vehicle matrix. Furthermore, since the vehicle is yet to arrive to the VC, no task is assigned to this vehicle.

The simulation will run until the sum of all the remaining task execution times is zero. During the run, we will find the event that will occur next by comparing the arrival time of the next vehicle, the shortest residency times of the vehicles in the VC, the shortest remaining execution time of the unfinished tasks. The comparison also allows us to determine the time delta until the next event (delta_t). Regardless of the type of the event, the interarrival time, the residency times of all the vehicles in the VC and the execution times of the assigned tasks must be subtracted by delta_t. Furthermore, the completion time of all unfinished tasks will increase by delta_t.

The type of the event will determine which complementary action will take place next. For instance, when the next event is an arrival, we need to check first if there are still unassigned

tasks. If that is indeed the case, then we have to randomly select one of the unassigned tasks and assign it to the arriving vehicle. Otherwise, the vehicle will change its status to "idle" in the VC upon arriving. When the next event is a completion of a task, then the status of the task will change to "complete". Additionally, the vehicle completing the task will continue to handle an unassigned task, if any. Finally, when the next event is a departure of the vehicle, then the status of the vehicle in the matrix will change. Furthermore, when the vehicle has been serving a task, the VC will assign the task to another idle vehicle, if any. Otherwise, the status of the task will change to suspend.

When a simulation ends, the completion time of the tasks should be available in the 3rd column of the task matrix. The VC completes a job when the remaining execution time of the last unfinished task of the job is zero. Therefore, the completion time of the job is the maximum value in the completion time column of the task matrix. We then transpose the column into a row and append the values to the completion time matrix.

Finally, the simulation is computed over 10^6 times to form the final completion matrix. Then from the matrix, we can compute the average and the distribution of the task completion time, the completion time of the j^{th} task, and the completion time of a job.

Appendix C

Analysis of job completion time in dynamic VC on a highway with congested traffic

As we mentioned previously, the analysis of the job completion time in dynamic VC on a highway with congested traffic shares a lot of methodological similarities with free-flowing traffic case. Hence, the section was distilled to show only important figures and results, and this appendix section aims to provide a more detailed analysis of the work.

Notation	Description	Notation	Description
$\Im$	Number of tasks in a job	$\mathcal{L}_{j,k}(s)$	Laplace transform of $P_{j,k}(t)$
μ	Service rate of a task	λ	Arrival rate of vehicles
α	Service rate of a vehicle	$\bar{\bar{r}}$	Average delay of a vehicle
ρ	Utilization factor of an $M/M/m$ queue	$\bar{Y}_n$	Average completion time of $n'th$ task
X_j	Absorbing time of subsystem θ_j to make the transition to an absorbing state	$g_{X_j}(x_j)$	Probability density function of the absorbing time X_j
k	Number of vehicles in the system, $k \geq 0$	$\mathcal{G}_j(s)$	Laplace transform of the probability density function of the absorbing time X_j
j	Number of uncompleted tasks in the system, $0 \leq j \leq \Im$	Y_n	Completion time of the $n'th$ task
$P_j(t)$	Probability that there are j uncompleted tasks in the subsystem θ_j at time t.	$\mathcal{Y}_n(s)$	Laplace transform of the probability density function of completion time of the $n'th$ task
$P_{j,k}(t)$	Probability that there are j uncompleted tasks and k vehicles in the subsystem θ_j at time t	$G_{X_j}(x_j)$	Cumulative distribution function of the absorbing time X_j
Q_k	Steady-state probability that there are k vehicles in the system	$\mathfrak{L}$	Average number of customers in the system of an $M/M/m$ queue
θ_j	Subsystem j representing the set of states that is reachable from state $\{j,k\}$ for a fixed value of j		

Table C.1. Main notations of the analysis of section 2.2

Vehicle model preliminary

Under congested traffic, we assume that the vehicle model follows the $M/M/m$ queuing model where m is the number of lanes on the highway. This change from $M/M/\infty$ allows us to incorporate the increased wait time of a vehicle in a VC when congestion occurs. We can find a similar assumption made under a single-lane scenario with a general service distribution in [122]. Vehicles arrive at the traffic at each lane according to the Poisson process with rate λ and spend an exponentially distributed time of mean α in the service to escape the traffic. Vehicles need to wait to be served with an average waiting time w. From [121], the distribution of vehicles in the system is given as follows:

$$\begin{aligned} Q_k &= Prob(k\ worker\ vehicles\ in\ the\ VC) \\ &= \begin{cases} Q_0 \dfrac{(m\rho)^k}{k!}, & k < m \\ Q_0 \dfrac{\rho^k m^m}{m!}, & k \geq m \end{cases} \end{aligned} \tag{C.1}$$

where the probability that there are 0 vehicles in the system is:

$$Q_0 = \left[\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \left(\frac{(m\rho)^m}{m!} \right) \left(\frac{1}{1-\rho} \right) \right]^{-1} \tag{C.2}$$

From [128, p. 42], the average number of customers in an $M/M/m$ queue, $\mathfrak{Q}$, is given by,

$$\mathfrak{Q} = \frac{\rho}{1-\rho} C(m, \lambda/\mu) + c\rho \tag{C.3}$$

where $\rho = \lambda/m\alpha$ is the utilization factor and $C(m, \lambda/\mu)$ is the Erlang C formula [121].

From Little's result average delay of a customer, $\bar{r}$, is given by,

$$\bar{r} = \frac{\mathfrak{Q}}{\lambda} \tag{C.4}$$

The utilization factor is ρ where it must follow the condition to be ergodic:

$$\rho = \frac{\lambda}{m\alpha} < 1 \tag{C.5}$$

Derivation of the joint probability distribution of the number of uncompleted tasks and vehicles

Similar to the analysis of the job completion time during a free-flow traffic scenario, a set of two variables, $\{j, k\}$, may represent the state of the system, where j denotes the number of uncompleted tasks, and k denotes the number of worker vehicles in the system at any time. We again note that when the system is in the state $\{j, k\}$, it means that the execution of $\Im - j$ tasks finishes, and the number of suspended tasks is given by $\max(0, j - k)$. Then, as shown in the state transition diagram in Fig. C.1, we can model the system using a two-dimensional birth-death process. However, when we write the global balance equation for this system, we will also end up with a differential-difference equation containing unknowns similar to that of equation (A.10).

In the state transition diagram of Fig. C.1, the states in row j correspond to having j uncompleted tasks in the system. The execution of the job begins in one of the states at the top row. Each time execution of a task is completed, the system moves to the next row below. The execution of the job is completed when the system enters one of the states at the bottom row. Thus the job completion time is the sum of the times the system spends in the states at each row. Following this observation, we will derive the amount of time the system spends in the states of each row. As a result, we will divide the system into several subsystems, where we can analyze each subsystem independently of the other subsystems. Let θ_j denote subsystem j where $0 < j \leq \Im$. We define subsystem θ_j as the set of states of rows j and $j - 1$ of the state-transition diagram as shown in Fig. C.2.

$$\theta_j = \{(j, k) \; \forall \; k \geq 0 \; \cup (j - 1, k) \; \forall \; k > 0\}, \qquad 0 < j \leq \Im \tag{C.6}$$

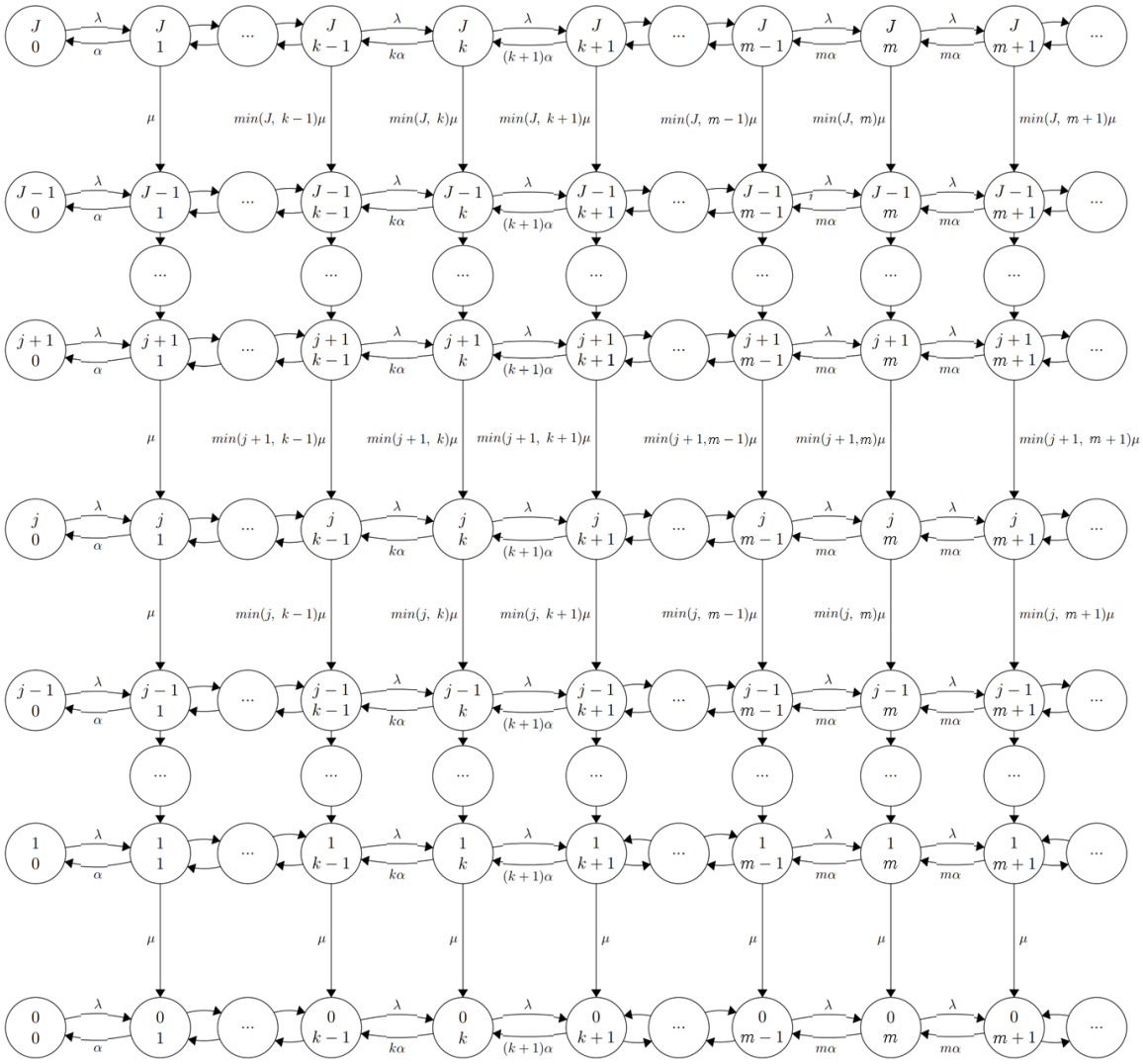


Fig. C.1. State transition diagram for the system in a congested traffic vehicle model. In each state number of uncompleted tasks is shown above the number of worker vehicles in the system.

The states of row $j-1$ for $k>0$ will be absorbing states [121] for this subsystem. When the subsystem enters one of the absorbing states, then the system immediately exits the subsystem θ_j and enters the subsystem θ_{j-1}. During this transition, the service of one of the tasks completes. We will refer to the amount of time that the system spends in the subsystem θ_j as absorbing time and denote it by X_j.

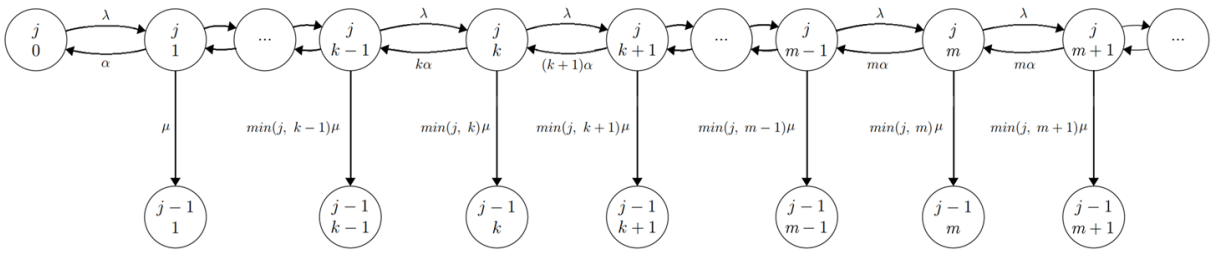


Fig. C.2. State transition diagram for sub system θ_j. States $\{j-1, k\}$ are absorption states. Once transitions to states $\{j-1, k\}$ occur, the reverse is not allowed.

Next, we will derive the joint probability distribution of the number of uncompleted tasks and the number of vehicles in a subsystem as a function of time. Since subsystems are analyzed independently, we assume that the initial time for each subsystem will be set to zero. Let $P_{j,k}(t)$ denote the probability that there will be j uncompleted tasks and k worker vehicles in the subsystem θ_j at time t. From Fig. C.2, the behavior of the subsystem θ_j may be described by the following set of differential-difference equations:

$$\frac{dP_{j,k}(t)}{dt} = \lambda P_{j,k-1}(t) + (k+1)\alpha P_{j,k+1}(t) - [\lambda + k\alpha + min(j,k)\mu]P_{j,k}(t), \quad 0 < k \leq m-1 \tag{C.7}$$

$$\frac{dP_{j,k}(t)}{dt} = \lambda P_{j,k-1}(t) + m\alpha P_{j,k+1}(t) - [\lambda + m\alpha + min(j,k)\mu]P_{j,k}(t), \qquad 0 < m \leq k \tag{C.8}$$

$$\frac{dP_{j-1,k}(t)}{dt} = min(j,k)\mu P_{j,k}(t), \qquad 0 < k \tag{C.9}$$

$$\frac{dP_{j,0}(t)}{dt} = \alpha P_{j,1}(t) - \lambda P_{j,0}(t), \qquad k = 0 \tag{C.10}$$

Since the number of worker vehicles in the system is at a steady state, then initial distribution of the number of workers in the subsystem $\theta_{\mathfrak{J}}$ is given by (C.1) and (C.2). In subsystem θ_j, $0 < j < \mathfrak{J}$, the initial number of vehicles in the system will be given by the number of vehicles in the subsystem θ_{j+1} when that subsystem has entered into an absorbing state. As a result, we have the following initial distributions,

$$P_{j,k}(0) = \begin{cases} Q_k, & for\ k \geq 0\ and\ \theta_j, j = \Im \\ P_{j+1,k}(\infty), & for\ \ \theta_j, j < \Im \end{cases} \tag{C.11}$$

We note that from the final value theorem property of the Laplace transforms,

$$P_{j+1,k}(\infty) = \lim_{s \to 0} s\mathcal{L}_{j+1,k}(s) \tag{C.12}$$

which gives the initial distribution of the number of vehicles in the subsystem θ_j. This initial distribution of the number of vehicles in a subsystem has also been confirmed by simulation.

Moreover, we also note that at time $t = 0$, no task should have been completed in the subsystem θ_j, thus, we can write,

$$P_{j-1,k}(0) = 0, \qquad 0 < j \leq \Im, \qquad k > 0, \tag{C.13}$$

Next, let us define the following Laplace transform,

$$\mathcal{L}_{j,k}(s) = \mathbb{L}\{P_{j,k}(t)\} = \int_{t=0}^{\infty} P_{j,k}(t)\ e^{-st} dt \tag{C.14}$$

To solve the set of equations (C.7) to (C.10), we will take their Laplace transforms,

$$s\mathcal{L}_{j,k}(s) - P_{j,k}(0) = \lambda\mathcal{L}_{j,k-1}(s) + (k+1)\alpha\mathcal{L}_{j,k+1}(s) - [\lambda + k\alpha + min(j,k)\mu]\mathcal{L}_{j,k}(s),$$

$$0 < k \leq m - 1 \tag{C.15}$$

$$s\mathcal{L}_{j,k}(s) - P_{j,k}(0) = \lambda\mathcal{L}_{j,k-1}(s) + m\alpha\mathcal{L}_{j,k+1}(s) - [\lambda + m\alpha + min(j,k)\mu]\mathcal{L}_{j,k}(s),$$

$$0 < m \leq k \tag{C.16}$$

$$s\mathcal{L}_{j-1,k}(s) - P_{j-1,k}(0) = min(j,k)\mu\mathcal{L}_{j,k}(s), \qquad k > 0 \tag{C.17}$$

$$s\mathcal{L}_{j,0}(s) - P_{j,0}(0) = \alpha\mathcal{L}_{j,1}(s) - \lambda\mathcal{L}_{j,0}(s), \qquad k = 0 \tag{C.18}$$

Next, we express $\mathcal{L}_{j,k}(s)$ in terms of $\mathcal{L}_{j,0}(s)$. From (C.18), we can write $\mathcal{L}_{j,1}(s)$ in terms of

$\mathcal{L}_{j,0}(s)$,

$$\mathcal{L}_{j,1}(s) = \frac{1}{\alpha}[(s+\lambda)\mathcal{L}_{j,0}(s) - P_{j,0}(0)] \tag{C.19}$$

Depending on the value of m, substituting $k = 1$ in either (C.15) or (C.16) gives, respectively,

$$\mathcal{L}_{j,2}(s) = \frac{[s+\lambda+\alpha+\mu]\mathcal{L}_{j,1}(s) - \lambda\mathcal{L}_{j,0}(s) - P_{j,1}(0)}{2\alpha},$$

$$for\ 0 < k \leq m-1 \tag{C.20}$$

$$\mathcal{L}_{j,2}(s) = \frac{[s+\lambda+m\alpha+\mu]\mathcal{L}_{j,1}(s) - \lambda\mathcal{L}_{j,0}(s) - P_{j,1}(0)}{m\alpha},$$

$$for\ 0 < m \leq k \tag{C.21}$$

Substituting (C.19) in either (C.20) or (C.21), we can also express $L_{j,2}(s)$ in terms of $L_{j,0}(s)$. Next, we solve for $L_{j,k+1}(s)$ in (C.15) and (C.16), respectively,

$$\mathcal{L}_{j,k+1}(s) = \frac{[s+\lambda+k\alpha+min(j,k)\mu]}{(k+1)\alpha}\mathcal{L}_{j,k}(s) - \frac{\lambda}{(k+1)\alpha}\mathcal{L}_{j,k-1}(s) - \frac{1}{(k+1)\alpha}P_{j,k}(0), \qquad 0 < k \leq m-1 \tag{C.22}$$

$$\mathcal{L}_{j,k+1}(s) = \frac{[s+\lambda+m\alpha+min(j,k)\mu]}{m\alpha}\mathcal{L}_{j,k}(s) - \frac{\lambda}{m\alpha}\mathcal{L}_{j,k-1}(s) - \frac{1}{m\alpha}P_{j,k}(0), \qquad 0 < m \leq k \tag{C.23}$$

In the above, we see that $\mathcal{L}_{j,k+1}(s)$ depends on $\mathcal{L}_{j,k}(s)$, and $\mathcal{L}_{j,k-1}(s)$. As a result, we can express $\mathcal{L}_{j,k+1}(s)$ recursively in terms of $\mathcal{L}_{j,0}(s)$ using (C.19), (C.20), and (C.21). This recursion implies $\mathcal{L}_{j,k}(s)$ can be written as a function of $\mathcal{L}_{j,0}(s)$ for $k > 0$. Next, we show how to determine $\mathcal{L}_{j,0}(s)$ from the normalization condition,

$$\sum_{k=0}^{\infty} P_{j,k}(t) + \sum_{k=1}^{\infty} P_{j-1,k}(t) = 1 \tag{C.24}$$

Taking Laplace transform of the conservation relation in the above, we have,

$$\sum_{k=0}^{\infty} \mathcal{L}_{j,k}(s) + \sum_{k=1}^{\infty} \mathcal{L}_{j-1,k}(s) = \frac{1}{s} \tag{C.25}$$

After substituting (C.13) in (C.17), we solve for $\mathcal{L}_{j-1,k}(s)$, then this result is substituted in (C.25), which gives,

$$s\mathcal{L}_{j,0}(s) + \sum_{k=1}^{\infty} [s + min(j,k)\mu]\mathcal{L}_{j,k}(s) = 1 \tag{C.26}$$

Next, since $\mathcal{L}_{j,k}(s)$ can be written as a function of $\mathcal{L}_{j,0}(s)$, from (C.22) and (C.23), we can substitute it into (C.26) to solve for the unknown function $\mathcal{L}_{j,0}(s)$. After determination of $\mathcal{L}_{j,0}(s)$, it means that we have obtained all $\mathcal{L}_{j,k}(s)$ functions and then application of the inverse Laplace transforms to $\mathcal{L}_{j,k}(s)$ results in $P_{j,k}(t)$. In determining $\mathcal{L}_{j,0}(s)$ we need to truncate the infinite summation in (C.26), the accuracy of this truncation is tested through simulation.

Derivation of the probability density function of the job completion time and the number of service interruptions in congested traffic

Since we also determined $P_{j,k}(t)$ for the congested traffic vehicle model, the derivation of the probability density function of the job completion time, and the number of service interruptions in congested traffic are similar to that of free-flow traffic and will not be given here.

Appendix D

Proof of equation (3.9)

We will first need to prove an important result to derive equation (3.9), which is

$$\int \alpha e^{-\alpha x} \frac{e^{-k\alpha x}(e^{\alpha x}-1)^k}{k!} dx = \frac{e^{-(k+1)\alpha x}(e^{\alpha x}-1)^{k+1}}{(k+1)!} \tag{D.1}$$

We prove (D.1) by taking the derivative with respect to x on the right-hand side. Denote $v(x) = \frac{e^{-(k+1)\alpha x}(e^{\alpha x}-1)^{k+1}}{(k+1)!}$,

$$\frac{dv(x)}{dx} = \frac{1}{(k+1)!} \cdot \frac{d}{dx}\left[e^{-(k+1)\alpha x}(e^{\alpha x}-1)^{k+1}\right]$$

$$= \frac{1}{(k+1)!}\left\{\frac{d}{dx}\left[e^{-(k+1)\alpha x}\right](e^{\alpha x}-1)^{k+1} + e^{-(k+1)\alpha x}\frac{d}{dx}[(e^{\alpha x}-1)^{k+1}]\right\}$$

$$= \frac{1}{(k+1)!}\left[(k+1)\alpha e^{-(k+1)\alpha x}(e^{\alpha x}-1)^k e^{\alpha x} - (k+1)\alpha e^{-(k+1)\alpha x}(e^{\alpha x}-1)^k(e^{\alpha x}-1)\right]$$

$$= \frac{\alpha e^{-\alpha x}e^{-k\alpha x}(e^{\alpha x}-1)^k}{k!} \tag{D.2}$$

Since the residency times or the workers are i.i.d and exponentially distributed, their joint pdf can be written as

$$g_{R_{(K)},\dots,R_{(1)}}(r_K,\dots,r_1) = K!\,\alpha^K e^{-\alpha r_1}\dots e^{-\alpha r_K} \tag{D.3}$$

Then we can rewrite the integral in equation (3.9) as below

$$\int_0^{r_{K-\Im+1}} \dots \int_0^{r_2} g_{R_{(K)},\dots,R_{(1)}}(r_K,\dots,r_1)dr_1\dots dr_{K-\Im} \tag{D.4}$$

$$= K!\,\alpha^{\Im} e^{-\alpha(r_K+\cdots+r_{K-\Im+1})} \cdot \int_0^{r_{K-\Im+1}} \alpha e^{-\alpha r_{K-\Im}} \dots \int_0^{r_2} \alpha e^{-\alpha r_1} dr_1 \dots dr_{K-\Im}$$

After that, we can apply result of (D.1) to (D.4) starting from $k = 0$ in (D.1) since each definite integral in (D.4) has the lower limit of 0 and the upper limit of its following integral. We finally have,

$$\int_0^{r_{K-\Im+1}} \dots \int_0^{r_2} g_{R_{(K)},\dots,R_{(1)}}(r_K, \dots, r_1) dr_1 \dots dr_{K-\Im}$$

$$= K!\,\alpha^{\Im} e^{-\alpha(r_K+\cdots+r_{K-\Im+1})} \cdot \frac{e^{-(K-J)\alpha r_{K-J+1}} (e^{\alpha r_{K-J+1}} - 1)^{K-J}}{(K-J)!} \tag{D.5}$$

Appendix E

Chapter 3 simulation description

We use Monte Carlo simulation to validate our numerical results since we employed stochastic modeling as our analytical methods.

At the start of a simulation run, a task matrix of size $2 \times \mathfrak{J}$ is created. The first row of the matrix is an ascending order array of $\mathfrak{J}$ exponential random variables with mean $1/\mu$, which is created by sorting the exprnd function with parameters $1/\mu$, 1 and $\mathfrak{J}$. The second row is the zero-value array to flag completed tasks. Then, we initiate a counter variable, veh_count, with value zero to keep track of the number of vehicles arrival to assign all tasks in a job in a simulation run. Additionally, another arrival time variable, veh_arrival_curr, is initiate with value zero to accumulate the interarrival times of all vehicles arrival at the VC. Essentially, when a vehicle arrives, veh_arrival_curr will increase the amount of interarrival time of the vehicle. Furthermore, if the vehicle can execute a task, then veh_arrival_curr also allows us to register the task completion time by adding the task execution time to veh_arrival_curr.

To register ordered task completion times, we initiate a zero-value array of $\mathfrak{J}$ elements. Thus, there will be a $n \times \mathfrak{J}$ matrix to record the results of the simulation runs. Additionally, we will also create two arrays of n elements to store the job completion time and the number of vehicles arrival to assign all tasks in a job for each simulation run.

In each simulation run, we will loop until the second row of the task matrix is all flagged after initiating all the necessary variables above. During each iteration, we will increment veh_count and generate two exponential random variables of the interarrival time and residency time of a vehicle using exprnd with the parameter $1/\lambda$ and $1/\alpha$, respectively. Then, veh_arrival_curr increases by the recently generated interarrival time. After that, we compare the residency time to the first row of the task matrix to see if the residency time is larger than any uncompleted task. If so, we will flag the task with the longest execution time. Then we compute the task completion time by adding the residency time to veh_arrival_curr and appending it to the task completion time array with the corresponding index. Table E.1 shows an example of an

arriving vehicle with the residency time of 4.96 compared to the tasks. We see that the residency time is larger than values of the first row of the matrix at index 0, 1, 2, 3, 4 and indices 1 and 4 is flagged at the second row. Since the task row is sorted in the ascending order, task at index 3 is the largest among the uncompleted tasks. Thus, the index 3 at the second row will be flagged. On the other hand, if the residency time is smaller than all the incomplete tasks, we will let the loop continue to run and do nothing.

Current vehicle's residency time: 4.96							
	Tasks						
	0	1	2	3	4	…	$\mathfrak{J}$
Ordered Execution time	1.57	2.31	2.95	4.88	4.92	…	10.34
Completion flag		x		x	x	…	

Table E.1. An example of the comparison of the residency time of a vehicle to the task matrix to determine which task to assign to the vehicle.

After all the tasks are flagged, we should have the array of the task completion time in the second row of the task matrix and the number of vehicle arrivals to assign all tasks in a job. The array is appended to the ordered task completion time. Then, the job completion time, which is the longest task completion time, is then appended to the array of job completion time. Finally, the number of vehicle arrivals is recorded for the run. The matrix and the arrays will allow us to compute the average and the distribution of the j^{th} ordered task completion times, the job completion time and the number of vehicle arrivals to assign all the tasks.

Finally the simulation is ran over 10^5 times to obtain the final results.

REFERENCES

[1] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An Overview on Edge Computing Research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020, doi: 10.1109/ACCESS.2020.2991734.

[2] A. Iqbal and S. Olariu, "A Survey of Enabling Technologies for Smart Communities," *Smart Cities*, vol. 4, no. 1, pp. 54–77, Dec. 2020, doi: 10.3390/smartcities4010004.

[3] L. Liu *et al.*, "Computing Systems for Autonomous Driving: State of the Art and Challenges," *IEEE Internet Things J*, vol. 8, no. 8, pp. 6469–6486, Apr. 2021, doi: 10.1109/JIOT.2020.3043716.

[4] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review," *Sensors*, vol. 21, no. 6, p. 2140, Mar. 2021, doi: 10.3390/s21062140.

[5] Y. Zhang, H. Zhang, K. Long, Q. Zheng, and X. Xie, "Software-Defined and Fog-Computing-Based Next Generation Vehicular Networks," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 34–41, Sep. 2018, doi: 10.1109/MCOM.2018.1701320.

[6] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular Fog Computing: A Viewpoint of Vehicles as the Infrastructures," *IEEE Trans Veh Technol*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016, doi: 10.1109/TVT.2016.2532863.

[7] A. M. A. Hamdi, F. K. Hussain, and O. K. Hussain, "Task offloading in vehicular fog computing: State-of-the-art and open issues," *Future Generation Computer Systems*, vol. 133, pp. 201–212, Aug. 2022, doi: 10.1016/j.future.2022.03.019.

[8] J. Lu, B. Li, H. Li, and A. Al-Barakani, "Expansion of city scale, traffic modes, traffic congestion, and air pollution," *Cities*, vol. 108, p. 102974, Jan. 2021, doi: 10.1016/j.cities.2020.102974.

[9] I. Gokasar, A. Timurogullari, M. Deveci, and H. Garg, "SWSCAV: Real-time traffic management using connected autonomous vehicles," *ISA Trans*, vol. 132, pp. 24–38, Jan. 2023, doi: 10.1016/j.isatra.2022.06.025.

[10] Y. Yang, L. Zhang, Y. Zhao, K.-K. R. Choo, and Y. Zhang, "Privacy-Preserving Aggregation-Authentication Scheme for Safety Warning System in Fog-Cloud Based VANET," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 317–331, 2022, doi: 10.1109/TIFS.2022.3140657.

[11] M. Benadda and G. Belalem, "Improving Road Safety for Driver Malaise and Sleepiness Behind the Wheel Using Vehicular Cloud Computing and Body Area Networks," *International Journal of Software Science and Computational Intelligence*, vol. 12, no. 4, pp. 19–41, Oct. 2020, doi: 10.4018/IJSSCI.2020100102.

[12] NHTSA, "Newly Released Estimates Show Traffic Fatalities Reached a 16-Year High in 2021," *NHTSA*, May 22, 2022.

[13] L. Nkenyereye, S. M. R. Islam, M. Bilal, M. Abdullah-Al-Wadud, A. Alamri, and A. Nayyar, “Secure crowd-sensing protocol for fog-based vehicular cloud,” *Future Generation Computer Systems*, vol. 120, pp. 61–75, Jul. 2021, doi: 10.1016/j.future.2021.02.008.

[14] A. Paranjothi, M. Atiquzzaman, and M. S. Khan, “Message Dissemination in Connected Vehicles,” Sep. 2020.

[15] M. Al Shabi, “An Efficient Delay Aware Emergency Message,” Jul. 2022.

[16] W. Yang, X. Dai, J. Xiao, and H. Jin, “LDV: A Lightweight DAG-Based Blockchain for Vehicular Social Networks,” *IEEE Trans Veh Technol*, vol. 69, no. 6, pp. 5749–5759, Jun. 2020, doi: 10.1109/TVT.2020.2963906.

[17] R. S. and C. Chetanaprakash, “Advancement in infotainment system in automotive sector with vehicular cloud network and current state of art,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 2, p. 2077, Apr. 2020, doi: 10.11591/ijece.v10i2.pp2077-2087.

[18] N. Gaouar and M. Lehsaini, “Toward vehicular cloud/fog communication: A survey on data dissemination in vehicular ad hoc networks using vehicular cloud/fog computing,” *International Journal of Communication Systems*, vol. 34, no. 13, Sep. 2021, doi: 10.1002/dac.4906.

[19] H. Singh, V. Laxmi, A. Malik, and Isha, “Fog Computing as Future Perspective in Vehicular Ad hoc Networks,” in *Fog, Edge, and Pervasive Computing in Intelligent IoT Driven Applications*, Wiley, 2020, pp. 177–192. doi: 10.1002/9781119670087.ch10.

[20] W. Wei, R. Yang, H. Gu, W. Zhao, C. Chen, and S. Wan, “Multi-Objective Optimization for Resource Allocation in Vehicular Cloud Computing Networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25536–25545, Dec. 2022, doi: 10.1109/TITS.2021.3091321.

[21] A. U. Rahman, A. W. Malik, V. Sati, A. Chopra, and S. D. Ravana, “Context-aware opportunistic computing in vehicle-to-vehicle networks,” *Vehicular Communications*, vol. 24, p. 100236, Aug. 2020, doi: 10.1016/j.vehcom.2020.100236.

[22] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, “A survey on vehicular cloud computing,” *Journal of Network and Computer Applications*, vol. 40, pp. 325–344, Apr. 2014, doi: 10.1016/j.jnca.2013.08.004.

[23] H. Qi and A. Gani, “Research on mobile cloud computing: Review, trend and perspectives,” in *2012 Second International Conference on Digital Information and Communication Technology and it’s Applications (DICTAP)*, IEEE, May 2012, pp. 195–202. doi: 10.1109/DICTAP.2012.6215350.

[24] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge Computing: Vision and Challenges,” *IEEE Internet Things J*, vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[25] L. M. Vaquero and L. Rodero-Merino, “Finding your Way in the Fog,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, Oct. 2014, doi: 10.1145/2677046.2677052.

[26] Y.-J. Ku, P.-H. Chiang, and S. Dey, “Quality of Service Optimization for Vehicular Edge Computing with Solar-Powered Road Side Units,” in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, IEEE, Jul. 2018, pp. 1–10. doi: 10.1109/ICCCN.2018.8487353.

[27] M. Khayyat, I. A. Elgendy, A. Muthanna, A. S. Alshahrani, S. Alharbi, and A. Koucheryavy, “Advanced Deep Learning-Based Computational Offloading for Multilevel Vehicular Edge-Cloud Computing Networks,” *IEEE Access*, vol. 8, pp. 137052–137062, 2020, doi: 10.1109/ACCESS.2020.3011705.

[28] S. Xu, C. Guo, R. Q. Hu, and Y. Qian, “Blockchain-Inspired Secure Computation Offloading in a Vehicular Cloud Network,” *IEEE Internet Things J*, vol. 9, no. 16, pp. 14723–14740, Aug. 2022, doi: 10.1109/JIOT.2021.3054866.

[29] E. Qafzezi, K. Bylykbashi, M. Ikeda, K. Matsuo, and L. Barolli, “Coordination and management of cloud, fog and edge resources in SDN-VANETs using fuzzy logic: A comparison study for two fuzzy-based systems,” *Internet of Things*, vol. 11, p. 100169, Sep. 2020, doi: 10.1016/j.iot.2020.100169.

[30] T. Mekki, I. Jabri, A. Rachedi, and M. Ben Jemaa, “Towards Multi-Access Edge Based Vehicular Fog Computing Architecture,” in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, Dec. 2018, pp. 1–6. doi: 10.1109/GLOCOM.2018.8647850.

[31] Z. Zhou, H. Liao, X. Wang, S. Mumtaz, and J. Rodriguez, “When Vehicular Fog Computing Meets Autonomous Driving: Computational Resource Management and Task Offloading,” *IEEE Netw*, vol. 34, no. 6, pp. 70–76, Nov. 2020, doi: 10.1109/MNET.001.1900527.

[32] S.-S. Lee and S. Lee, “Resource Allocation for Vehicular Fog Computing Using Reinforcement Learning Combined With Heuristic Information,” *IEEE Internet Things J*, vol. 7, no. 10, pp. 10450–10464, Oct. 2020, doi: 10.1109/JIOT.2020.2996213.

[33] R. Kumar, A. Saad, and R. E. De Grande, “COrRect: Connection-Oriented Resource Matching for Vehicular Clouds,” in *IEEE International Conference on Communications*, Institute of Electrical and Electronics Engineers Inc., Jun. 2021. doi: 10.1109/ICC42927.2021.9500563.

[34] Y. Wu, J. Wu, L. Chen, G. Zhou, and J. Yan, “Fog Computing Model and Efficient Algorithms for Directional Vehicle Mobility in Vehicular Network,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 5, pp. 2599–2614, May 2021, doi: 10.1109/TITS.2020.2971343.

[35] H. Sami, A. Mourad, and W. El-Hajj, “Vehicular-OBUs-As-On-Demand-Fogs: Resource and Context Aware Deployment of Containerized Micro-Services,” *IEEE/ACM*

Transactions on Networking, vol. 28, no. 2, pp. 778–790, Apr. 2020, doi: 10.1109/TNET.2020.2973800.

[36] L. H. Yen, J. C. Hu, Y. D. Lin, and B. Kar, "Decentralized Configuration Protocols for Low-Cost Offloading from Multiple Edges to Multiple Vehicular Fogs," *IEEE Trans Veh Technol*, vol. 70, no. 1, pp. 872–885, Jan. 2021, doi: 10.1109/TVT.2020.3046874.

[37] F. Sun *et al.*, "Cooperative Task Scheduling for Computation Offloading in Vehicular Cloud," *IEEE Trans Veh Technol*, vol. 67, no. 11, pp. 11049–11061, Nov. 2018, doi: 10.1109/TVT.2018.2868013.

[38] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-Latency Tradeoff for Dynamic Computation Offloading in Vehicular Fog Computing," *IEEE Trans Veh Technol*, vol. 69, no. 12, pp. 14198–14211, Dec. 2020, doi: 10.1109/TVT.2020.3040596.

[39] Z. Liu, P. Dai, H. Xing, Z. Yu, and W. Zhang, "A Distributed Algorithm for Task Offloading in Vehicular Networks With Hybrid Fog/Cloud Computing," *IEEE Trans Syst Man Cybern Syst*, vol. 52, no. 7, pp. 4388–4401, Jul. 2022, doi: 10.1109/TSMC.2021.3097005.

[40] S. Xu and C. Guo, "Computation Offloading in a Cognitive Vehicular Networks with Vehicular Cloud Computing and Remote Cloud Computing," *Sensors*, vol. 20, no. 23, p. 6820, Nov. 2020, doi: 10.3390/s20236820.

[41] A. Waheed, M. A. Shah, A. Khan, C. Maple, and I. Ullah, "Hybrid Task Coordination Using Multi-Hop Communication in Volunteer Computing-Based VANETs," *Sensors*, vol. 21, no. 8, p. 2718, Apr. 2021, doi: 10.3390/s21082718.

[42] Y. Kang, Z. Liu, Q. Chen, and Y. Dai, "Joint Task Offloading and Resource Allocation Strategy for DiffServ in Vehicular Cloud System," *Wirel Commun Mob Comput*, vol. 2020, pp. 1–15, Nov. 2020, doi: 10.1155/2020/8823173.

[43] S. Ucar, T. Higuchi, C.-H. Wang, and O. Altintas, "Chain of Interdependent Vehicular Micro Clouds," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, IEEE, Apr. 2021, pp. 1–5. doi: 10.1109/VTC2021-Spring51267.2021.9448911.

[44] S. K. Pande *et al.*, "A Smart Cloud Service Management Algorithm for Vehicular Clouds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5329–5340, Aug. 2021, doi: 10.1109/TITS.2020.3021075.

[45] Q. Chen, Y. Xie, S. Guo, J. Bai, and Q. Shu, "Sensing system of environmental perception technologies for driverless vehicle: A review of state of the art and challenges," *Sens Actuators A Phys*, vol. 319, p. 112566, Mar. 2021, doi: 10.1016/j.sna.2021.112566.

[46] Z. Ning *et al.*, "Joint Computing and Caching in 5G-Envisioned Internet of Vehicles: A Deep Reinforcement Learning-Based Traffic Control System," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5201–5212, Aug. 2021, doi: 10.1109/TITS.2020.2970276.

[47] E. Benalia, S. Bitam, and A. Mellouk, “Data dissemination for Internet of vehicle based on 5G communications: A survey,” *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 5, May 2020, doi: 10.1002/ett.3881.

[48] W. Anwar, N. Franchi, and G. Fettweis, “Physical Layer Evaluation of V2X Communications Technologies: 5G NR-V2X, LTE-V2X, IEEE 802.11bd, and IEEE 802.11p,” in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, IEEE, Sep. 2019, pp. 1–7. doi: 10.1109/VTCFall.2019.8891313.

[49] S. Gyawali, S. Xu, Y. Qian, and R. Q. Hu, “Challenges and Solutions for Cellular Based V2X Communications,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 222–255, Oct. 2021, doi: 10.1109/COMST.2020.3029723.

[50] S. Maaloul, H. Aniss, L. Mendiboure, and M. Berbineau, “Performance Analysis of Existing ITS Technologies: Evaluation and Coexistence,” *Sensors*, vol. 22, no. 24, p. 9570, Dec. 2022, doi: 10.3390/s22249570.

[51] M. Hao, D. Ye, S. Wang, B. Tan, and R. Yu, “URLLC Resource Slicing and Scheduling in 5G Vehicular Edge Computing,” in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, IEEE, Apr. 2021, pp. 1–5. doi: 10.1109/VTC2021-Spring51267.2021.9448805.

[52] H. Zhang, Z. Wang, and K. Liu, “V2X offloading and resource allocation in SDN-assisted MEC-based vehicular networks,” *China Communications*, vol. 17, no. 5, pp. 266–283, May 2020, doi: 10.23919/JCC.2020.05.020.

[53] H. Xiao, W. Zhang, W. Li, A. T. Chronopoulos, and Z. Zhang, “Joint Clustering and Blockchain for Real-Time Information Security Transmission at the Crossroads in C-V2X Networks,” *IEEE Internet Things J*, vol. 8, no. 18, pp. 13926–13938, Sep. 2021, doi: 10.1109/JIOT.2021.3068175.

[54] J. Lu, W. Yang, and F. Wu, “High Definition Map Distribution in Named Data Networking Based VANETs,” in *2020 3rd International Conference on Hot Information-Centric Networking (HotICN)*, IEEE, Dec. 2020, pp. 129–134. doi: 10.1109/HotICN50779.2020.9350807.

[55] L. M. Clements and K. M. Kockelman, “Economic Effects of Automated Vehicles,” *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2606, no. 1, pp. 106–114, Jan. 2017, doi: 10.3141/2606-14.

[56] K. Nisar *et al.*, “A survey on the architecture, application, and security of software defined networking: Challenges and open issues,” *Internet of Things*, vol. 12, p. 100289, Dec. 2020, doi: 10.1016/j.iot.2020.100289.

[57] D. Jiang, Z. Wang, L. Huo, and S. Xie, “A Performance Measurement and Analysis Method for Software-Defined Networking of IoV,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3707–3719, Jun. 2021, doi: 10.1109/TITS.2020.3029076.

[58] R. Amin, I. Pali, and V. Sureshkumar, “Software-Defined Network enabled Vehicle to Vehicle secured data transmission protocol in VANETs,” *Journal of Information Security and Applications*, vol. 58, p. 102729, May 2021, doi: 10.1016/j.jisa.2020.102729.

[59] A. H. Sodhro, J. J. P. C. Rodrigues, S. Pirbhulal, N. Zahid, A. R. L. de Macedo, and V. H. C. de Albuquerque, “Link Optimization in Software Defined IoV Driven Autonomous Transportation System,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3511–3520, Jun. 2021, doi: 10.1109/TITS.2020.2973878.

[60] A. Abuarqoub, “A Review of the Control Plane Scalability Approaches in Software Defined Networking,” *Future Internet*, vol. 12, no. 3, p. 49, Mar. 2020, doi: 10.3390/fi12030049.

[61] H. Goumidi, S. Harous, Z. Aliouat, and A. M. Gueroui, “Lightweight Secure Authentication and Key Distribution Scheme for Vehicular Cloud Computing,” *Symmetry (Basel)*, vol. 13, no. 3, p. 484, Mar. 2021, doi: 10.3390/sym13030484.

[62] V. Kumar, M. Ahmad, D. Mishra, S. Kumari, and M. K. Khan, “RSEAP: RFID based secure and efficient authentication protocol for vehicular cloud computing,” *Vehicular Communications*, vol. 22, p. 100213, Apr. 2020, doi: 10.1016/j.vehcom.2019.100213.

[63] X. Shen, Y. Lu, Y. Zhang, X. Liu, and L. Zhang, “An Innovative Data Integrity Verification Scheme in the Internet of Things assisted information exchange in transportation systems,” *Cluster Comput*, vol. 25, no. 3, pp. 1791–1803, Jun. 2022, doi: 10.1007/s10586-021-03471-5.

[64] A. Masood, D. S. Lakew, and S. Cho, “Security and Privacy Challenges in Connected Vehicular Cloud Computing,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2725–2764, Feb. 2020, doi: 10.1109/COMST.2020.3012961.

[65] P. Mundhe, S. Verma, and S. Venkatesan, “A comprehensive survey on authentication and privacy-preserving schemes in VANETs,” *Comput Sci Rev*, vol. 41, p. 100411, Aug. 2021, doi: 10.1016/j.cosrev.2021.100411.

[66] S. Wang, J. Li, G. Wu, H. Chen, and S. Sun, “Joint Optimization of Task Offloading and Resource Allocation Based on Differential Privacy in Vehicular Edge Computing,” *IEEE Trans Comput Soc Syst*, vol. 9, no. 1, pp. 109–119, Feb. 2022, doi: 10.1109/TCSS.2021.3074949.

[67] Y. Li, H. Li, G. Xu, T. Xiang, and R. Lu, “Practical Privacy-Preserving Federated Learning in Vehicular Fog Computing,” *IEEE Trans Veh Technol*, vol. 71, no. 5, pp. 4692–4705, May 2022, doi: 10.1109/TVT.2022.3150806.

[68] M. Zamzam, T. El-Shabrawy, and M. Ashour, “Game Theory for Computation Offloading and Resource Allocation in Edge Computing: A Survey,” in *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, IEEE, Oct. 2020, pp. 47–53. doi: 10.1109/NILES50944.2020.9257921.

[69] A. Mebrek and A. Yassine, “Intelligent Resource Allocation and Task Offloading Model for IoT Applications in Fog Networks: A Game-Theoretic Approach,” *IEEE Trans Emerg Top Comput Intell*, pp. 1–15, 2021, doi: 10.1109/TETCI.2021.3102214.

[70] Z. Wang, S. Zheng, Q. Ge, and K. Li, “Online Offloading Scheduling and Resource Allocation Algorithms for Vehicular Edge Computing System,” *IEEE Access*, vol. 8, pp. 52428–52442, 2020, doi: 10.1109/ACCESS.2020.2981045.

[71] H. Wang, T. Lv, Z. Lin, and J. Zeng, “Energy-Delay Minimization of Task Migration Based on Game Theory in MEC-Assisted Vehicular Networks,” *IEEE Trans Veh Technol*, vol. 71, no. 8, pp. 8175–8188, Aug. 2022, doi: 10.1109/TVT.2022.3175238.

[72] X. Peng, K. Ota, and M. Dong, “Multiattribute-Based Double Auction Toward Resource Allocation in Vehicular Fog Computing,” *IEEE Internet Things J*, vol. 7, no. 4, pp. 3094–3103, Apr. 2020, doi: 10.1109/JIOT.2020.2965009.

[73] D. Kumar, G. Baranwal, and D. P. Vidyarthi, “A Survey on Auction based Approaches for Resource Allocation and Pricing in Emerging Edge Technologies,” *J Grid Comput*, vol. 20, no. 1, p. 3, Mar. 2022, doi: 10.1007/s10723-021-09593-9.

[74] J. Shyuan, W. Lim, Z. Xiong, T. D. Niyato, C. Leung, and C. Miao, “A Double Auction Mechanism for Resource Allocation in Coded Vehicular Edge Computing,” *IEEE Trans Veh Technol*, vol. 71, no. 2, pp. 1832–1845, Feb. 2022, doi: 10.1109/TVT.2021.3131395.

[75] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, “Computation Resource Allocation and Task Assignment Optimization in Vehicular Fog Computing: A Contract-Matching Approach,” *IEEE Trans Veh Technol*, vol. 68, no. 4, pp. 3113–3125, Apr. 2019, doi: 10.1109/TVT.2019.2894851.

[76] A. Alamer and S. Basudan, “An efficient truthfulness privacy-preserving tendering framework for vehicular fog computing,” *Eng Appl Artif Intell*, vol. 91, p. 103583, May 2020, doi: 10.1016/j.engappai.2020.103583.

[77] X. Zhu, Y. Luo, A. Liu, N. N. Xiong, M. Dong, and S. Zhang, “A Deep Reinforcement Learning-Based Resource Management Game in Vehicular Edge Computing,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2422–2433, Mar. 2022, doi: 10.1109/TITS.2021.3114295.

[78] S. M. A. Kazmi *et al.*, “Computing on Wheels: A Deep Reinforcement Learning-Based Approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 22535–22548, Nov. 2022, doi: 10.1109/TITS.2022.3165662.

[79] Y. Sun, S. Zhou, and Z. Niu, “Distributed Task Replication for Vehicular Edge Computing: Performance Analysis and Learning-Based Algorithm,” *IEEE Trans Wirel Commun*, vol. 20, no. 2, pp. 1138–1151, Feb. 2021, doi: 10.1109/TWC.2020.3030889.

[80] C. Zhu *et al.*, “Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing,” *IEEE Internet Things J*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019, doi: 10.1109/JIOT.2018.2875520.

[81] T. Bahreini, M. Brocanelli, and D. Grosu, “Energy-Aware Resource Management in Vehicular Edge Computing Systems,” in *2020 IEEE International Conference on Cloud Engineering (IC2E)*, IEEE, Apr. 2020, pp. 49–58. doi: 10.1109/IC2E48712.2020.00012.

[82] J. B. D. da Costa, R. I. Meneguette, D. Rosario, and L. A. Villas, “Combinatorial Optimization-based Task Allocation Mechanism for Vehicular Clouds,” in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, IEEE, May 2020, pp. 1–5. doi: 10.1109/VTC2020-Spring48590.2020.9128834.

[83] X. Xu, R. Gu, F. Dai, L. Qi, and S. Wan, “Multi-objective computation offloading for Internet of Vehicles in cloud-edge computing,” *Wireless Networks*, vol. 26, no. 3, pp. 1611–1629, Apr. 2020, doi: 10.1007/s11276-019-02127-y.

[84] H. Materwala, L. Ismail, R. M. Shubair, and R. Buyya, “Energy-SLA-aware genetic algorithm for edge–cloud integrated computation offloading in vehicular networks,” *Future Generation Computer Systems*, vol. 135, pp. 205–222, Oct. 2022, doi: 10.1016/j.future.2022.04.009.

[85] W. Shu and Y. Li, “Joint offloading strategy based on quantum particle swarm optimization for MEC-enabled vehicular networks,” *Digital Communications and Networks*, vol. 9, no. 1, pp. 56–66, Feb. 2023, doi: 10.1016/j.dcan.2022.03.009.

[86] N. Keshari, T. S. Gupta, and D. Singh, “Particle Swarm Optimization based Task Offloading in Vehicular Edge Computing,” in *2021 IEEE 18th India Council International Conference (INDICON)*, IEEE, Dec. 2021, pp. 1–8. doi: 10.1109/INDICON52576.2021.9691758.

[87] X. Liu and G. Zhang, “Joint Optimization Offloading and Resource Allocation in Vehicular Edge Cloud Computing Networks with Delay Constraints,” in *2020 IEEE International Conference on Progress in Informatics and Computing (PIC)*, IEEE, Dec. 2020, pp. 363–368. doi: 10.1109/PIC50277.2020.9350840.

[88] S. Midya, A. Roy, K. Majumder, and S. Phadikar, “Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach,” *Journal of Network and Computer Applications*, vol. 103, pp. 58–84, Feb. 2018, doi: 10.1016/j.jnca.2017.11.016.

[89] P. Lang, D. Tian, X. Duan, J. Zhou, Z. Sheng, and V. C. M. Leung, “Cooperative Computation Offloading in Blockchain-Based Vehicular Edge Computing Networks,” *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 783–798, Sep. 2022, doi: 10.1109/TIV.2022.3190308.

[90] Q. Hu, H. Cheng, X. Zhang, and C. Lin, “Trusted resource allocation based on proof-of-reputation consensus mechanism for edge computing,” *Peer Peer Netw Appl*, vol. 15, no. 1, pp. 444–460, Jan. 2022, doi: 10.1007/s12083-021-01240-0.

[91] X. Ye, M. Li, P. Si, R. Yang, Z. Wang, and Y. Zhang, “Collaborative and Intelligent Resource Optimization for Computing and Caching in IoV With Blockchain and MEC

Using A3C Approach," *IEEE Trans Veh Technol*, vol. 72, no. 2, pp. 1449–1463, Feb. 2023, doi: 10.1109/TVT.2022.3210570.

[92] M. V. Fard, A. Sahafi, A. M. Rahmani, and P. S. Mashhadi, "Resource allocation mechanisms in cloud computing: a systematic literature review," *IET Software*, vol. 14, no. 6, pp. 638–653, Dec. 2020, doi: 10.1049/iet-sen.2019.0338.

[93] C. Tran and M. Mehmet-Ali, "Analysis of Job Completion Time in Vehicular Cloud Under Concurrent Task Execution," in *2023 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, Feb. 2023, pp. 101–105. doi: 10.1109/ICNC57223.2023.10074524.

[94] C. Tran and M. Mehmet-Ali, "Towards Job Completion Time in Vehicular Cloud by Overcoming Resource Volatility," in *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, IEEE, Sep. 2022, pp. 331–334. doi: 10.1109/LCN53696.2022.9843398.

[95] B. Kar, K.-M. Shieh, Y.-C. Lai, Y.-D. Lin, and H.-W. Ferng, "QoS Violation Probability Minimization in Federating Vehicular-Fogs With Cloud and Edge Systems," *IEEE Trans Veh Technol*, vol. 70, no. 12, pp. 13270–13280, Dec. 2021, doi: 10.1109/TVT.2021.3120413.

[96] R. Fantacci and B. Picano, "Performance Analysis of a Delay Constrained Data Offloading Scheme in an Integrated Cloud-Fog-Edge Computing System," *IEEE Trans Veh Technol*, vol. 69, no. 10, pp. 12004–12014, Oct. 2020, doi: 10.1109/TVT.2020.3008926.

[97] Y. Katayama and T. Tachibana, "Optimal Task Allocation Algorithm Based on Queueing Theory for Future Internet Application in Mobile Edge Computing Platform," *Sensors*, vol. 22, no. 13, p. 4825, Jun. 2022, doi: 10.3390/s22134825.

[98] C. Celes, A. Boukerche, and A. A. F. Loureiro, "Revealing and Modeling Vehicular Micro Clouds Characteristics in a Large-Scale Mobility Trace," in *IEEE International Conference on Communications*, Institute of Electrical and Electronics Engineers Inc., Jun. 2021. doi: 10.1109/ICC42927.2021.9500262.

[99] A. Bonadio, F. Chiti, and R. Fantacci, "Performance Analysis of an Edge Computing SaaS System for Mobile Users," *IEEE Trans Veh Technol*, vol. 69, no. 2, pp. 2049–2057, Feb. 2020, doi: 10.1109/TVT.2019.2957938.

[100] Y. Wu and J. Zheng, "Modeling and Analysis of the Local Delay in an MEC-Based VANET for a Suburban Area," *IEEE Internet Things J*, vol. 9, no. 9, pp. 7065–7079, May 2022, doi: 10.1109/JIOT.2021.3116195.

[101] J. Li, X. You, and J. Zheng, "Performance Modeling and Analysis of anMECSystem with Task Priority and Expiring Time Constraint," *IEEE Communications Letters*, pp. 1–1, 2023, doi: 10.1109/LCOMM.2023.3270338.

[102] B. Ravi, A. Gautam, and J. Thangaraj, "Stochastic performance modeling and analysis of multi service provisioning with software defined vehicular networks," *AEU -*

International Journal of Electronics and Communications, vol. 124, p. 153327, Sep. 2020, doi: 10.1016/j.aeue.2020.153327.

[103] W. Miao, G. Min, X. Zhang, Z. Zhao, and J. Hu, "Performance Modelling and Quantitative Analysis of Vehicular Edge Computing With Bursty Task Arrivals," *IEEE Trans Mob Comput*, vol. 22, no. 2, pp. 1129–1142, Feb. 2023, doi: 10.1109/TMC.2021.3087013.

[104] J. Zhou, D. Tian, Y. Wang, Z. Sheng, X. Duan, and V. C. M. Leung, "Reliability-Optimal Cooperative Communication and Computing in Connected Vehicle Systems," *IEEE Trans Mob Comput*, vol. 19, no. 5, pp. 1216–1232, May 2020, doi: 10.1109/TMC.2019.2907491.

[105] P. Sun and N. Samaan, "A Novel VANET-Assisted Traffic Control for Supporting Vehicular Cloud Computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 6726–6736, Nov. 2021, doi: 10.1109/TITS.2020.2994280.

[106] Q. Wu, Y. Zhao, and Q. Fan, "Time-Dependent Performance Modeling for Platooning Communications at Intersection," *IEEE Internet Things J*, vol. 9, no. 19, pp. 18500–18513, Oct. 2022, doi: 10.1109/JIOT.2022.3161028.

[107] J. Kim and G. Hwang, "Performance Modeling and Analysis of Broadcast Packets in Vehicular Ad Hoc Networks," *J Syst Sci Syst Eng*, vol. 28, no. 2, pp. 211–223, Apr. 2019, doi: 10.1007/s11518-018-5397-1.

[108] R. Florin, P. Ghazizadeh, A. Ghazi Zadeh, S. El-Tawab, and S. Olariu, "Reasoning About Job Completion Time in Vehicular Clouds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1762–1771, Jul. 2017, doi: 10.1109/TITS.2016.2620434.

[109] R. Florin, P. Ghazizadeh, A. Ghazi Zadeh, R. Mukkamala, and S. Olariu, "A tight estimate of job completion time in vehicular clouds," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2018, doi: 10.1109/TCC.2018.2834352.

[110] R. Florin and S. Olariu, "Toward Approximating Job Completion Time in Vehicular Clouds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 8, pp. 3168–3177, Aug. 2019, doi: 10.1109/TITS.2018.2873998.

[111] R. Florin, A. Ghazizadeh, P. Ghazizadeh, S. Olariu, and D. C. Marinescu, "Enhancing Reliability and Availability through Redundancy in Vehicular Clouds," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 1061–1074, Jul. 2021, doi: 10.1109/TCC.2019.2905590.

[112] A. Ghazizadeh, P. Ghazizadeh, R. Mukkamala, and S. Olariu, "Towards Approximating Expected Job Completion Time in Dynamic Vehicular Clouds," in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, IEEE, Jul. 2019, pp. 481–483. doi: 10.1109/CLOUD.2019.00084.

[113] T. Bahreini, M. Brocanelli, and D. Grosu, “VECMAN: A Framework for Energy-Aware Resource Management in Vehicular Edge Computing Systems,” *IEEE Trans Mob Comput*, vol. 22, no. 2, pp. 1231–1245, Feb. 2023, doi: 10.1109/TMC.2021.3089338.

[114] M. Asim and A. A. Abd El-Latif, “Intelligent computational methods for multi-unmanned aerial vehicle-enabled autonomous mobile edge computing systems,” *ISA Trans*, vol. 132, pp. 5–15, Jan. 2023, doi: 10.1016/j.isatra.2021.11.021.

[115] R. Roess, E. Prassas, and W. McShane, *Traffic Engineering*. Pearson Prentice Hall, 2004.

[116] D. Bruneo, “A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 560–569, Mar. 2014, doi: 10.1109/TPDS.2013.67.

[117] E. Ataie, R. Entezari-Maleki, L. Rashidi, K. S. Trivedi, D. Ardagna, and A. Movaghar, “Hierarchical Stochastic Models for Performance, Availability, and Power Consumption Analysis of IaaS Clouds,” *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 1039–1056, Oct. 2019, doi: 10.1109/TCC.2017.2760836.

[118] R. Ghosh, F. Longo, R. Xia, V. K. Naik, and K. S. Trivedi, “Stochastic Model Driven Capacity Planning for an Infrastructure-as-a-Service Cloud,” *IEEE Trans Serv Comput*, vol. 7, no. 4, pp. 667–680, Oct. 2014, doi: 10.1109/TSC.2013.44.

[119] Y. Chen *et al.*, “Stochastic Workload Scheduling for Uncoordinated Datacenter Clouds with Multiple QoS Constraints,” *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1284–1295, Oct. 2020, doi: 10.1109/TCC.2016.2586048.

[120] S. Yousefi, E. Altman, R. El-Azouzi, and M. Fathy, “Analytical Model for Connectivity in Vehicular Ad Hoc Networks,” *IEEE Trans Veh Technol*, vol. 57, no. 6, pp. 3341–3356, Nov. 2008, doi: 10.1109/TVT.2008.2002957.

[121] L. Kleinrock, *Queueing systems*, vol. 1. Wiley-Interscience, 1975.

[122] T. Zhang, R. E. de Grande, and A. Boukerche, “Design and analysis of stochastic traffic flow models for vehicular clouds,” *Ad Hoc Networks*, vol. 52, pp. 39–49, Dec. 2016, doi: 10.1016/j.adhoc.2016.07.009.

[123] L. Liu and D.-H. Shi, “Busy period in $GI^X/G/\infty$,” *J Appl Probab*, vol. 33, no. 3, pp. 815–829, Sep. 1996, doi: 10.2307/3215361.

[124] D. N. Shanbhag, “On infinite server queues with batch arrivals,” *J Appl Probab*, vol. 3, no. 1, pp. 274–279, Jun. 1966, doi: 10.2307/3212053.

[125] J. A. Filipe and M. A. M. Ferreira, “Infinite servers queue systems busy period - a practical case on logistics problems solving,” *Applied Mathematical Sciences*, vol. 9, pp. 1221–1228, 2015, doi: 10.12988/ams.2015.410808.

[126] J. R. Artalejo and M. J. Lopez-Herrero, “Analysis of the busy period for the M/M/c queue: an algorithmic approach,” *J Appl Probab*, vol. 38, no. 01, pp. 209–222, Mar. 2001, doi: 10.1017/S0021900200018611.

[127] J. Wang, O. Baron, and A. Scheller-Wolf, "*M/M/c* Queue with Two Priority Classes," *Oper Res*, vol. 63, no. 3, pp. 733–749, Jun. 2015, doi: 10.1287/opre.2015.1375.

[128] M. Barbeau and E. Kranakis, *Principles of Ad Hoc Networking*. Chichester, UK: John Wiley & Sons, Ltd, 2007. doi: 10.1002/9780470512494.